WEB2-300 详细解题思路

Author:phithon <root@leavesongs.com>

拿到后台地址，不知道管理员账号、密码。有的同志想到社工、爆破之类的。其实依旧是找漏洞，我在 hint 里也说明了。

这一步需要深入 Codeigniter 核心框架。

浏览/xdsec_cms/core/Codeigniter.php，可以大概看出脚本执行流程：

core -> 实例化控制器（执行构造函数__construct） -> hook -> controller 主体函数

其中，hook 是脚本钩子，等于可以在执行的中途加入其它代码。

后台钩子的具体代码在/xdsec_app/admin_app/config/hooks.php

```
$hook['post_controller_constructor'] = function()
{
	$self = & get_instance();
	$self->load->library('session');
	if(SELF == "admin.php" || config_item("index_page") == "admin.php") {
		$self->error("Please rename admin filename 'admin.php' and config item 'index_page'", site_url());
	}
	$self->template_data["is_admin"] = $self->is_admin();
	if(method_exists($self, "init")) {
		call_user_func([$self, "init"]);
	}
};

$hook['post_controller'] = function()
{
	session_write_close();
};
```

我写了两个 hook，分别是 post_controller_constructor 和 post_controller。post_controller_constructor 是在控制器类实例化后，执行具体方法前，来执行。而且在 core 代码中，还有个点，如果我们实现了_remap 方法，那么_remap 方法也将 hook 掉原始的控制器方法：

```
if ( ! class_exists($class, FALSE) OR $method[0] === '_' OR method_exists('CI_Controller', $method))
{
    $e404 = TRUE;
}
elseif (method_exists($class, '_remap'))
{
    $params = array($method, array_slice($URI->rsegments, 2));
    $method = '_remap';
}
```

_remap 是在 $hook['post_controller_constructor'] 后执行的，我在 $hook['post_controller_constructor']中又定义了一个 init 方法，如果控制器中实现了这个方法将会调用之。

remap 方法我将其伪装成修改方法名的 hook 函数，实际上我在其中加入了一个 before_handler 方法，如果控制器实现了它，将会调用之。

（这两个方法实际上灵感来自 tornado，tornado 中就有这样的两个方法。）

代码在/xdsec_app/admin_app/core/Xdsec_Controller.php：

```
public function _remap($method, $params=[])
{
    $method = "handle_{$method}";
    if (method_exists($this, $method)) {
        if(method_exists($this, "before_handler")) {
            call_user_func([$this, "before_handler"]);
        }
        $ret = call_user_func_array([$this, $method], $params);
```

```
            if(method_exists($this, "after_handler")) {
                call_user_func([$this, "after_handler"]);
            }
            return $ret;
        } else {
            show_404();
        }
}
```

所以，综上所述，最后实际上整个脚本执行顺序是：
core -> __construct -> hook -> init -> before_hanlder（在此检查权限） -> controller 主体 -> after_handler

我将检查后台权限的代码放在 before_handler 中。而 init 方法的本意是初始化一些类变量。
但如果开发者错误地将关键代码放在了 init 方法或__construct 方法中，将造成一个越权。（因为还没执行检查权限的 before_handler 方法）

回到控制器代码中。/xdsec_app/admin_app/controllers/Log.php 其中就有 init 函数：

```
    public function init()
    {
        $ip = I("post.ip/s") ? I("post.ip/s") : $this->input->ip_address();
        $this->default_log = $this->query_log($ip);
        $this->ip_address = $ip;
    }
```

很明显其中包含关键逻辑$this->query_log($ip);
跟进 query_log 方法：

```
protected function query_log($value, $key="ip")
    {
```

```
        $user_table = $this->db->dbprefix("admin");
        $log_table = $this->db->dbprefix("adminlog");
        switch($key) {
            case "ip":
            case "time":
            case "log":
                $table = $log_table;
                break;
            case "username":
            case "aid":
            default:
                $table = $user_table;
                break;
        }
        $query = $this->db->query("SELECT `{$user_table}`.`username`,
`{$log_table}`.*
                                    FROM `{$user_table}`, `{$log_table}`
                                    WHERE `{$table}`.`{$key}`='{$value}'
                                    ORDER BY `{$log_table}`.`time` DESC
                                    LIMIT 20");
        if($query) {
            $ret = $query->result();
        } else {
            $ret = [];
        }
        return $ret;
    }
```

后台代码一般比前台代码安全性差，这里得到了很好的体现。后台大量 where 语句是直接拼接的字符串，我们看到这里将$value 拼接进了 SQL 语句。
而$value 即为$ip，$ip 可以来自$this->input->ip_address()。

熟悉 CI 的同学可能觉得没有问题，但其实我这里已经偷梁换柱得将 CI 自带的

ip_address 函数替换成我自己的了：
/xdsec_app/admin_app/core/Xdsec_Input.php

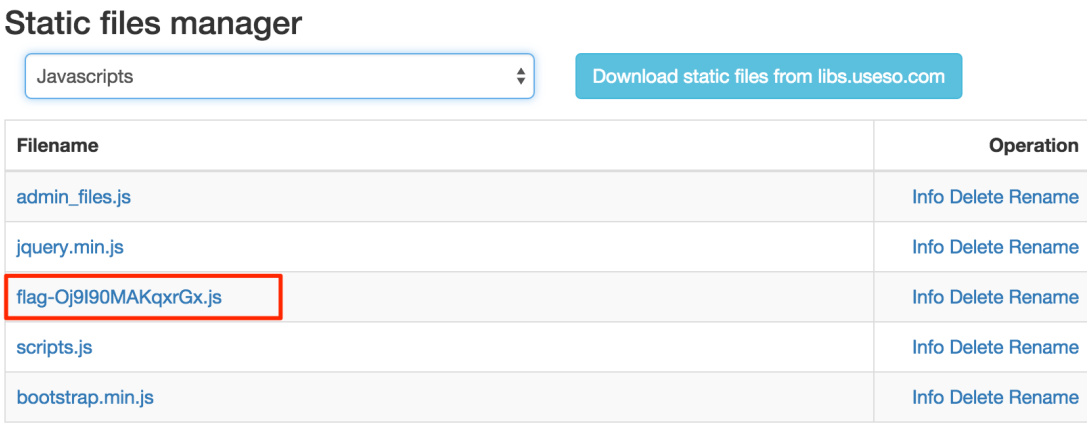
```
function ip_address()
{
    if (isset($_SERVER["HTTP_CLIENT_IP"])) {
        $ip = $_SERVER["HTTP_CLIENT_IP"];
    } elseif (isset($_SERVER["HTTP_X_FORWARDED_FOR"])) {
        $ip = $_SERVER["HTTP_X_FORWARDED_FOR"];
    } elseif (isset($_SERVER["REMOTE_ADDR"])) {
        $ip = $_SERVER["REMOTE_ADDR"];
    } else {
        $ip = CI_Input::ip_address();
    }
    if(!preg_match("/(\d+)\.(\d+)\.(\d+)\.(\d+)/", $ip))
        $ip = "127.0.0.1";
    return trim($ip);
}
```

这个函数看起来没有问题，实际上最后一个正则判断因为没有加^$定界符，所以形同虚设，只需利用“1.2.3.4’ union select …” 即可绕过。（这里的灵感来自我去年挖的 ThinkPHP 框架注入，也是没有首尾限定符，详见我乌云）
所以这里，结合上面说的 init 尚未检查权限的越权漏洞，组成一个无需后台登录的 SQL 注入。
但因为 init 后就是检查权限的函数，没有登录的情况下将会直接返回 302，而且后台数据库 debug 模式关闭了，无法报错。
这里只能利用 time-based 盲注。
多的不说，编写一个盲注脚本（xdseccms.py）即可跑出管理员密码：

跑出密码为：c983cff7bc504d350ede4758ab5a7b4b

cmd5 解密登录即可。

登录后台，在后台文件管理的 javascript 一项中发现第三个 flag：



这里说一下 ctf 技巧。

像我这种基于框架的代码审计，作者可能会修改框架核心代码（当然我这里没有，我都是正常 hook）。如果修改框架核心代码的话，就很不好找漏洞了，因为一般框架核心代码都比较多。

这时候你应该拿 diff 这类工具，把正常的框架和你下载的 ctf 源码进行比较，很

容易就能知道作者修改了哪些内容。