【漏洞分析】YXCMS SESSION 固定攻击

文/ icuke

0x00 漏洞概述

漏洞类型：会话固定攻击
危险等级：中
利用条件：该漏洞需要管理员在登录状态下访问攻击者构造的恶意链接。
漏洞位置：common.function.php 中的 session 方法
漏洞产生原因：session 方法允许 session_id()函数使用 request 接收的参数值设置会话 id
受影响版本：全部版本
修复版本：无

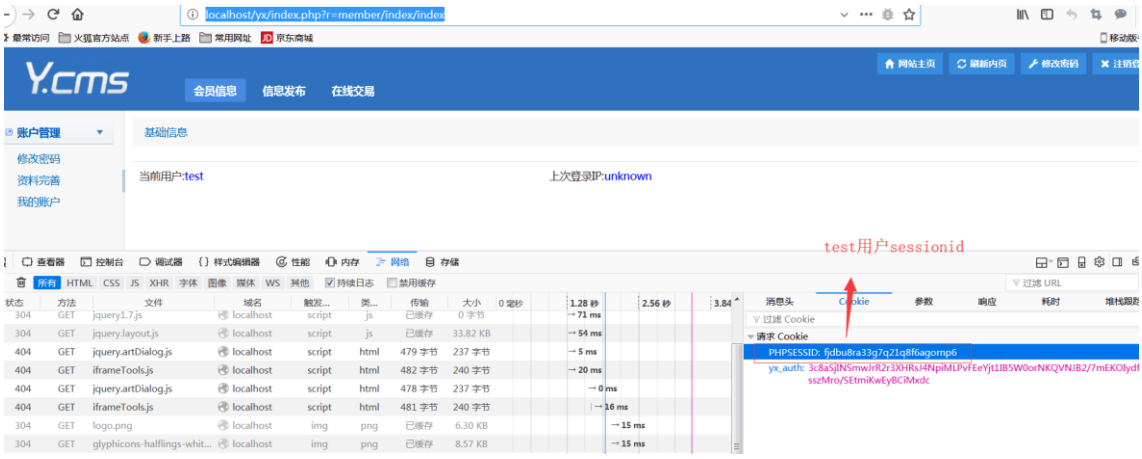
0x01 环境准备

IDE	PhpStorm
Web 环境	phpstudy 集成环境
PHP 版本	5.4.45
YXCMS 版本	1.4.7

0x02 漏洞复现

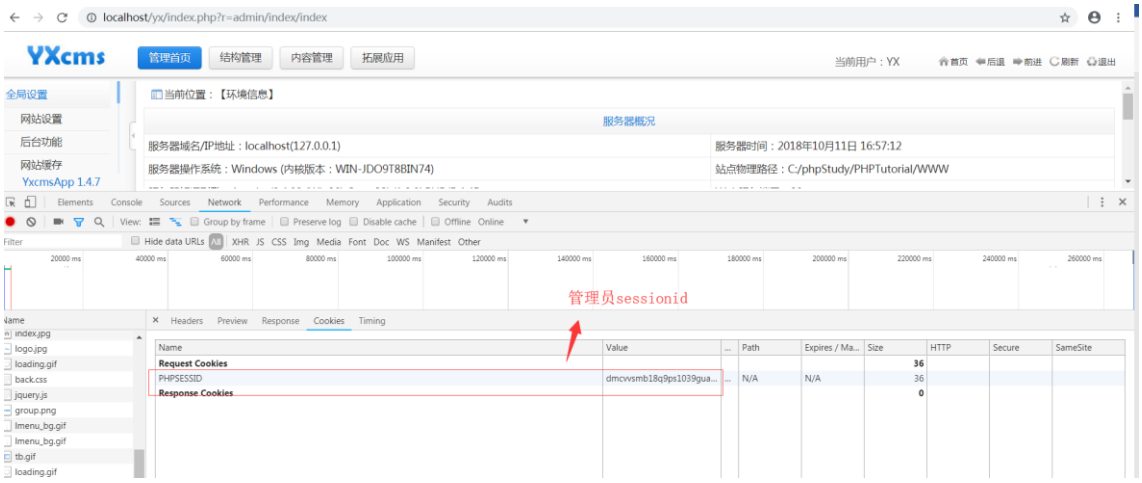
1. 模拟攻击者新注册一个普通用户

使用 firefox 访问 http://localhost/yx/index.php?r=member/index/regist，注册新用户，例 test，该用户可以视为攻击者账号。登录后获取 sessionid，此处为 fjdbu8ra33g7q21q8f6agornp6



2. 模拟受害者使用管理员账号登录后台

使用 chrome 访问 http://localhost/yx/index.php?r=admin/index/login 登录管理后台，YXCMS 默认账号密码为 admin:123456。正常管理员用户 sessionid 为 dmcvvsmb18q9ps1039guam77g6

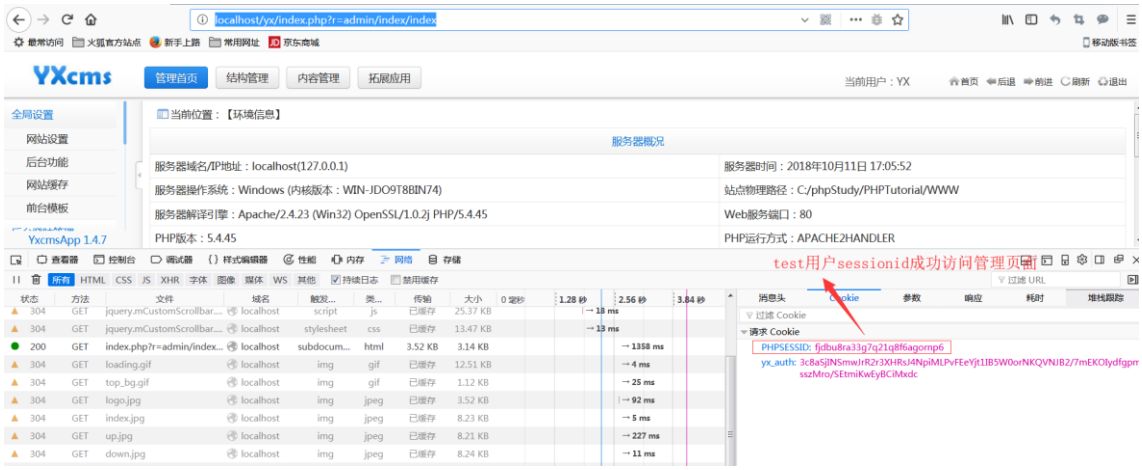


3. 模拟受害者访问恶意链接

受害者使用 chrome 访问恶意链接：
http://localhost/yx/index.php?r=admin/index/index&sessionid=fjdbu8ra33g7q21q8f6agornp6

sessionid 为上步攻击者用户的 sessionid。

攻击过程完成，攻击者使用 firefox 访问后台管理页面：
http://localhost/yx/index.php?r=admin/index/index
成功以管理员身份登录管理后台。



0x03 漏洞分析

1. 漏洞代码分析

漏洞代码在 /protected/include/lib/common.function.php 文件的 session 方法中

```php
function session($name='',$value = '') {
    if(empty($name)){
        return $_SESSION;
    }
    $sessionId = request( str: 'request.sessionid');
    if(!empty($sessionId)){
        session_id($sessionId);
    }
    if(!isset($_SESSION)){
        session_starts();
    }
    if($value === ''){
        $session = $_SESSION[$name];
    }else if($value==null){
        unset($_SESSION[$name]);
    }else{
        $session = $_SESSION[$name] = $value;
    }
    return $session;
}
```

存在风险代码

通过代码可知，如果 $sessionId 存在，则使用 session_id 函数将其设置为会话 id。$sessionId 由 request 方法得到。跟进 request 方法。

```php
668  function request($str, $default = null, $function = null) {
669      $str = trim($str);
670      list($method,$name) = explode( delimiter: '.',$str, limit: 2);
671      $method = strtoupper($method);
672      switch ($method) {
673          case 'POST':...
676          case 'SESSION':...
679          case 'REQUEST':
680              $type = $_REQUEST;
681              break;
682          case 'COOKIE':...
685          case 'GET':
686          default:
687              $type = $_GET;
688              break;
689      }
690      if(empty($name)){
691          $request = filter_string($type);
692      }else{
693          if($method == 'GET'){
694              $request = urldecode($type[$name]);
695          }else{
696              $request = $type[$name];
697          }
698          $request = filter_string($request);
699          //设置默认值
700          if($default){...}
705          //设置处理函数
706          if($function){...}
709      }
710      return $request;
711  }
```

传入 request 的 $str 参数值为 request.sessionid，第 670 行使用 explode 函数对传入的 $str 变量进行分割，分割后 $method= request，$name = sessionid。第 671 行将 request 转为 REQUEST，然后进入 switch 分支，执行代码语句 $type = $_REQUEST。

此时 $method = REQUEST，$name = sessionid，$type =$_REQUEST，代码执行到 696 行，$request = $type[$name]，变量替换后为 $request = $_REQUEST[sessionid]，由此可知 $request 变量值可通过 GET、POST、COOKIE 由用户远程传入。

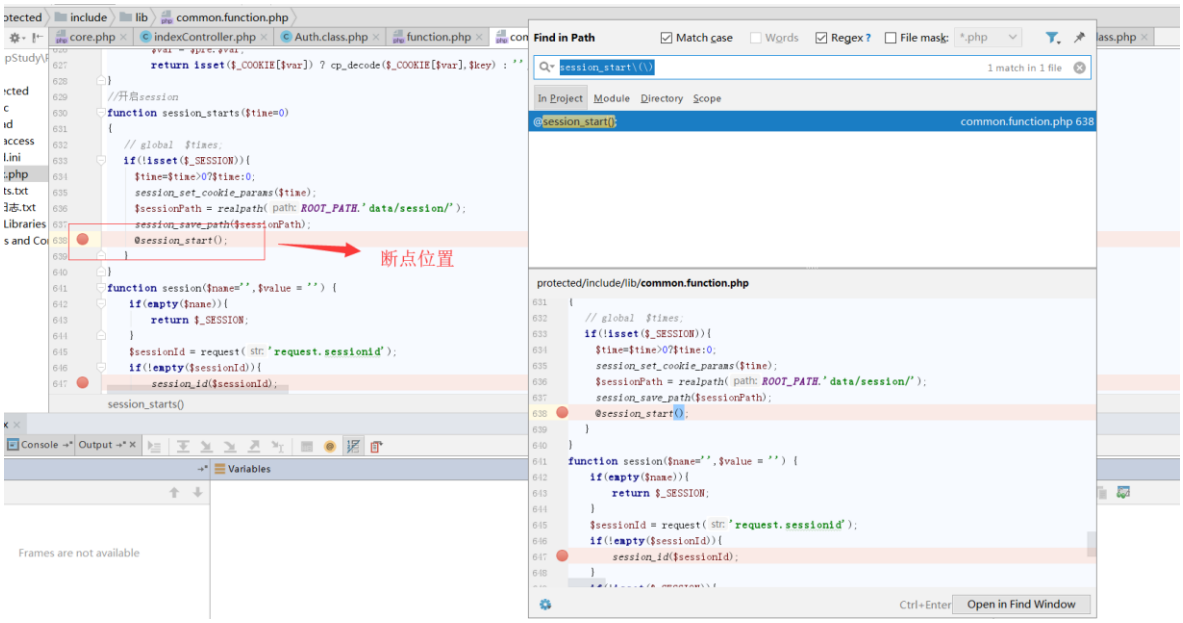
698 行使用自定义的 filter_string 方法对 $request 值做过滤，但由于传入的 sessionid 只含有数字字母，故此处可忽略。

最终此次 request 方法调用会返回一个由用户通过 GET、POST、COOKIE 传入的 sessionid 参数值。

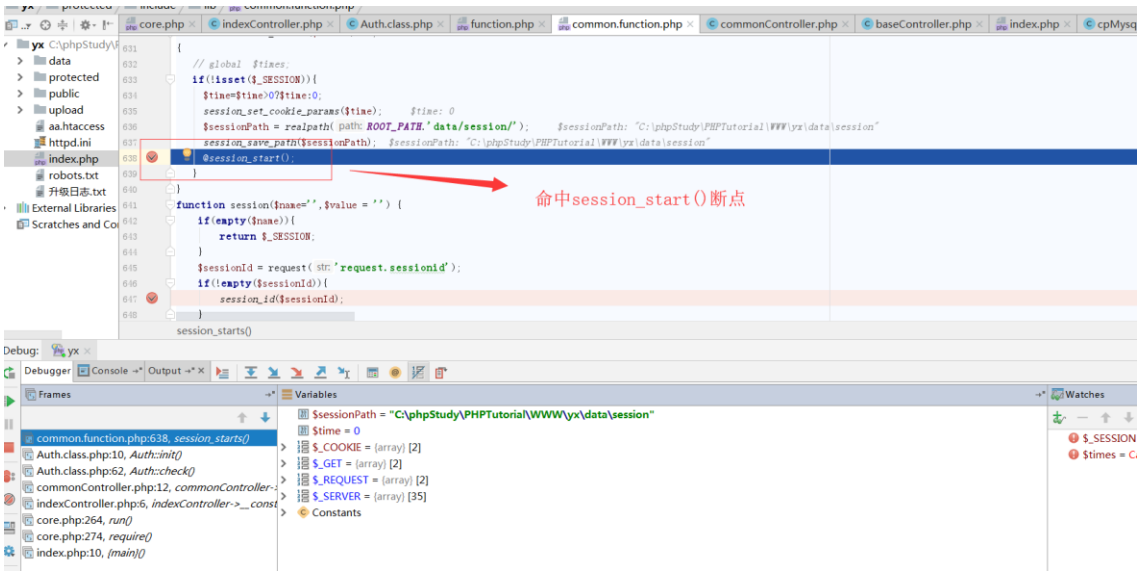
2. 回溯触发路径

此处使用 PhpStorm xdebug 调试方式进行动态调试。
session_id() 函数必须在 session_start() 函数后执行，如果想成功利用，调用顺序必须为 session_start() -> session_id(args)。session_id(args) 在 session 方法中，我们在此处下一个断点。

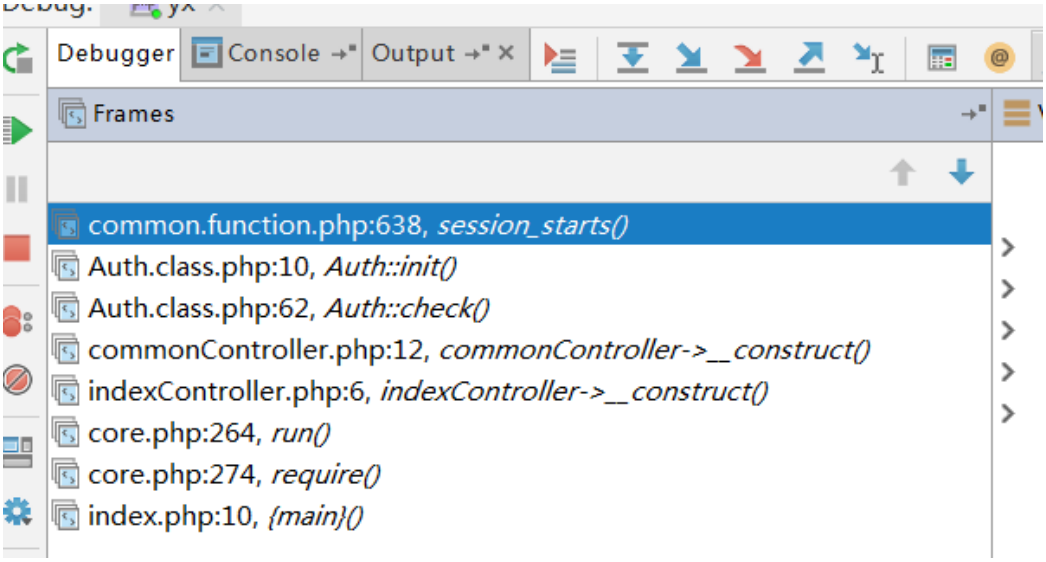
全局搜索一下 session_start()函数的调用方法，在 session_starts()方法中调用，设置断点。



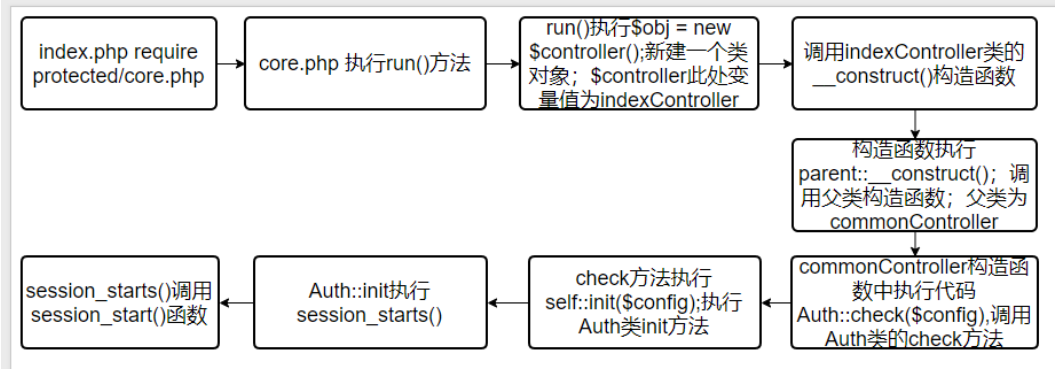
访问 http://localhost/yx/index.php?r=admin/index/index&sessionid=bbbbbbb&XDEBUG_SESSION_START=14527 查看调用关系，XDEBUG_SESSION_START 参数值由调试器生成。



左下方为调用过程：

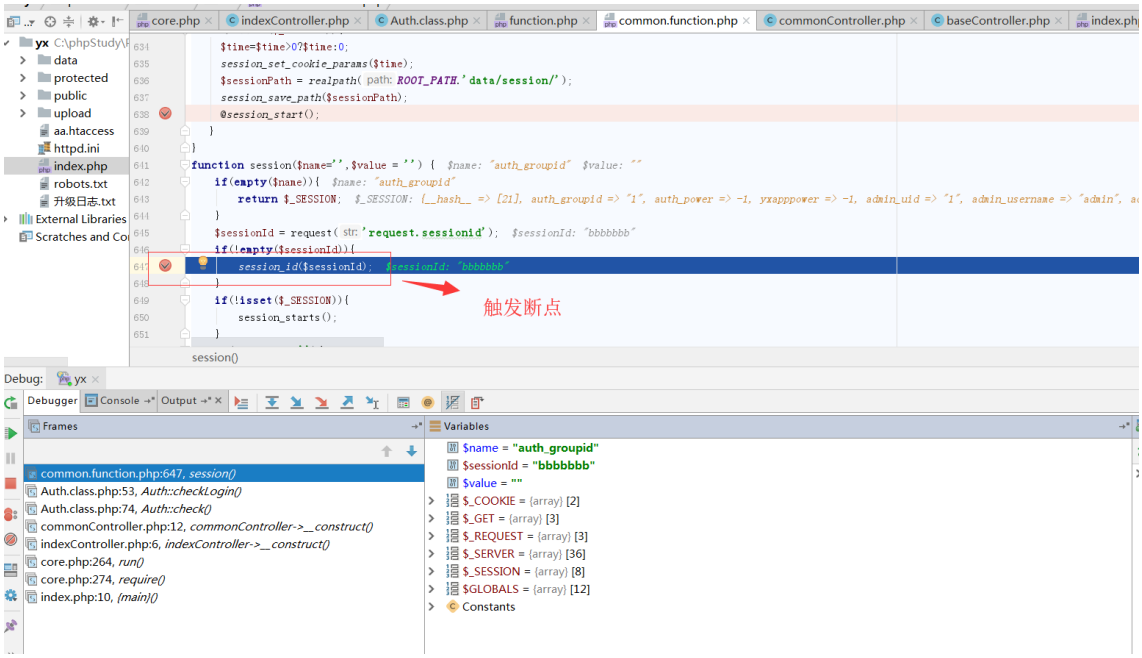


调用过程如下：

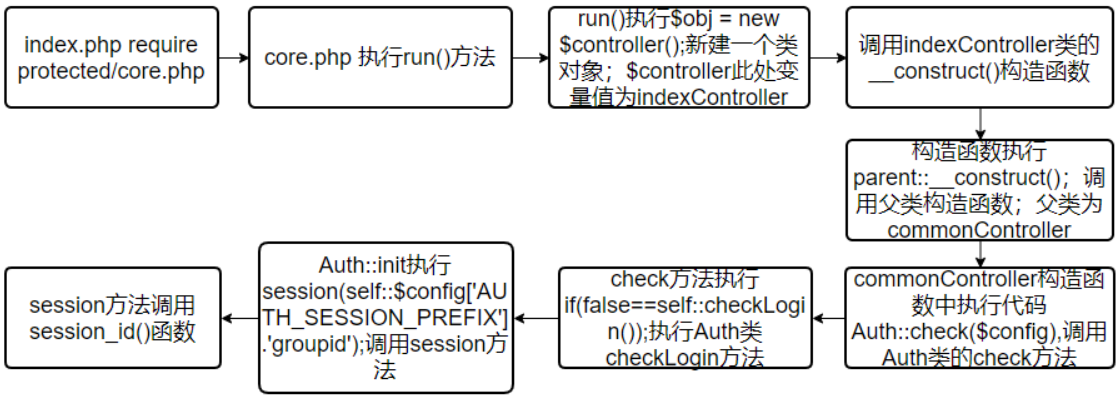


这里补充下 YXCMS 的路由寻址过程，我们传入的 url 为 http://localhost/index.php?r=admin/index/index，YXCMS 会调用 protected/core.php 的 urlRoute 方法对 r 参数值进行处理，提取出 app_name、controller_name、action_name，示例 url 处理后 app_name：admin、controller_name：index、action_name:index。所以图中第三步新建的类对象为 indexController 对象。

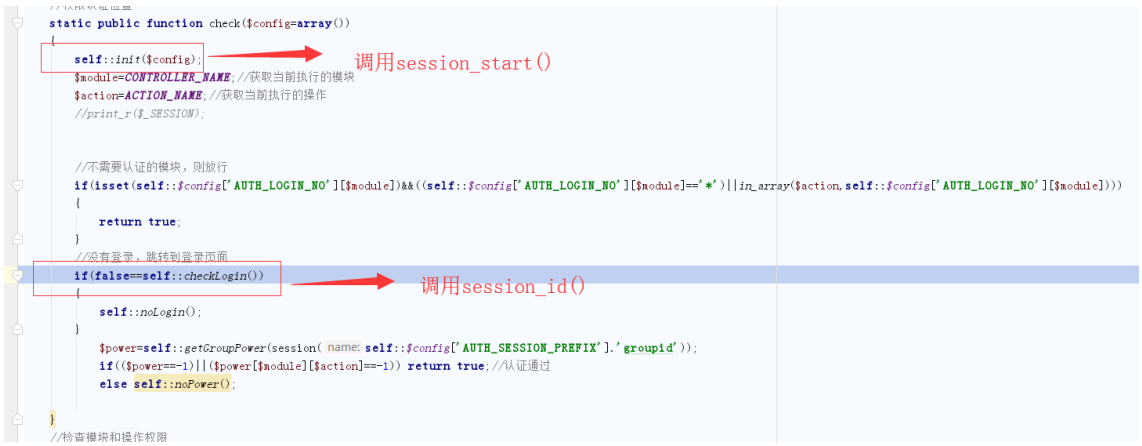
命中 session_start() 断点后按下 F9，跳转到下一个断点：



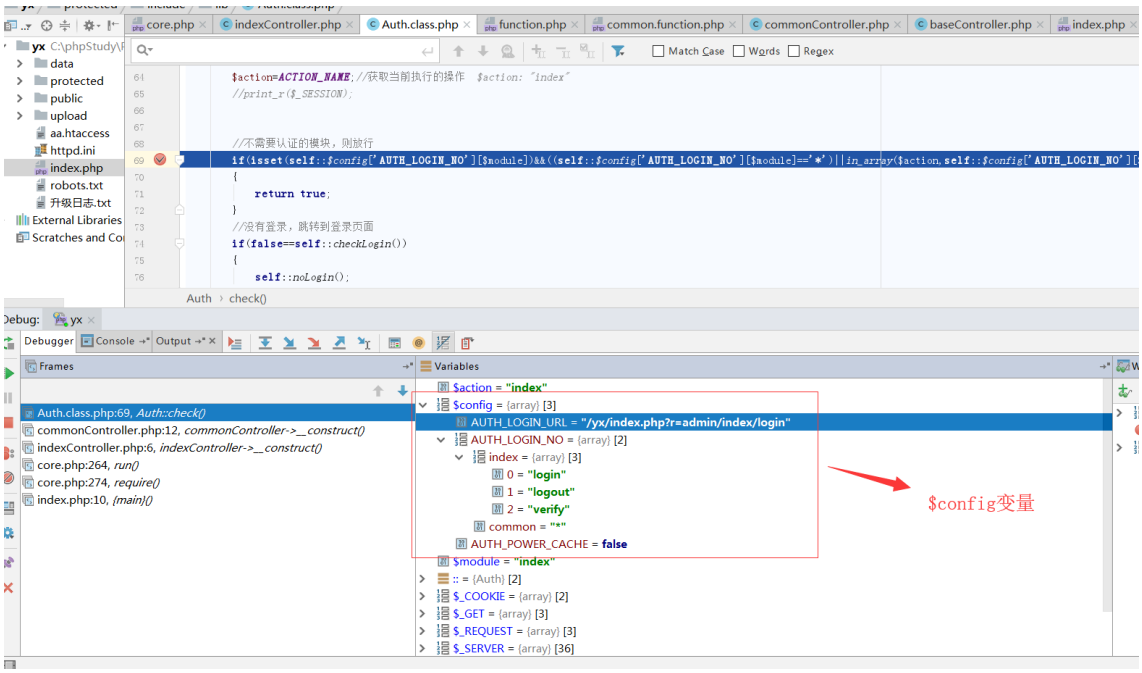
调用过程如下：



看一下命中 session_id() 的调用过程，和 session_start() 的调用过程相比，在 Auth::check($config) 方法后的调用出现差异，命中 session_start() 调用过程中会调用 Auth::init() 方法，session_id() 调用过程中会调用 Auth::checkLogin() 方法。看下 Auth::check 方法代码：



通过代码可知，如果要实现 session_start()->session_id() 流程，第 69 行的判断条件必须不能满足。通过注释可知此处是判断访问模块是否需要认证，不需要认证则直接返回，在此处设置一断点进行调试。



可以看到 $module 值为 index，$action 值也为 index，而判断条件中，只要 $module 不等于 common，$action 不为 login、logout、verify，代码会继续向下执行，执行 session_id() 函数。

0x04 总结

当管理员访问需要认证的模块时，会调用 Auth::init() 初始化一个 session 会话，即调用 session_start() 函数，然后会判断访问模块是否需要认证，若需要认证，则调用 session() 方法。如果此时在 GET、POST、COOKIE 中传入一个 sessionid，则会调用 session_id() 函数设置指定的会话 id，php 执行完成后，$_SESSION 变量中的内容会以序列化的形式写入到指定 session 文件中，使用此 sessionid 的普通用户就会获得管理员身份。

0x05 修复方法

删除 session_id() 调用代码