通达 OA 代码审计篇一：11.7 有条件的任意命令执行

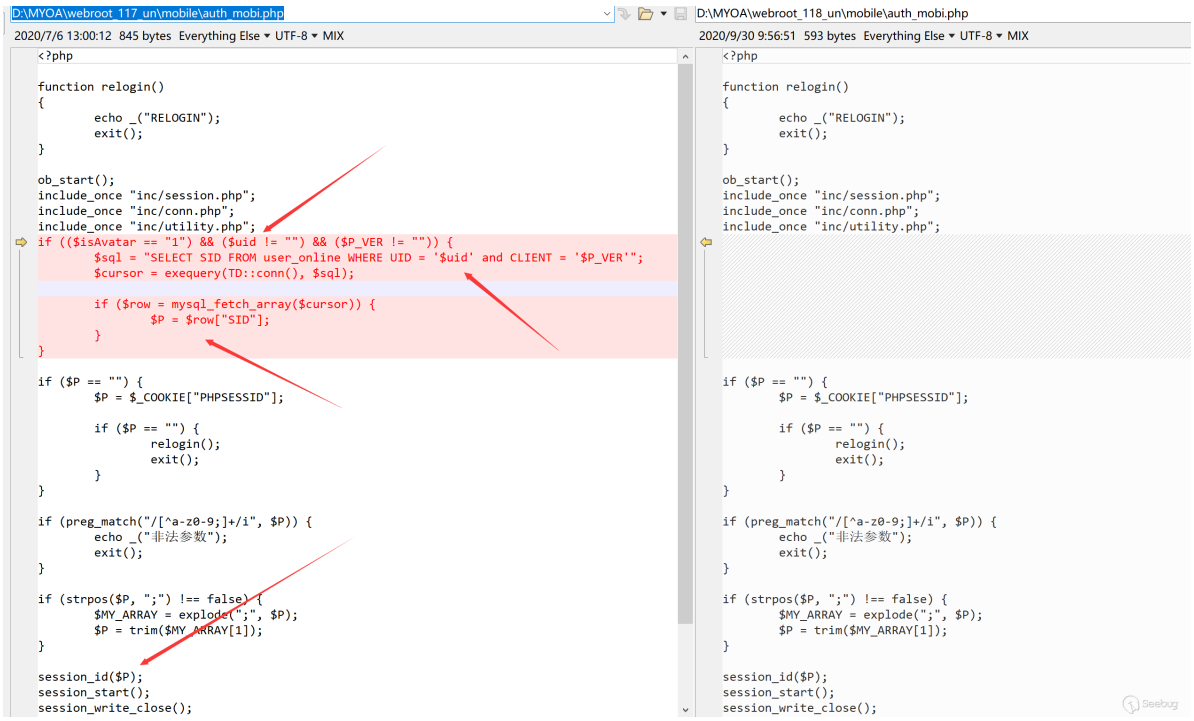
作者：LoRexxar'@知道创宇404实验室
日期：2021年3月3日

这是一个由有条件的任意用户登录+低权限文件上传+低权限目录穿越+低权限文件包含组成。可能是盯着国内OA的人太多了，这个漏洞在2020年9月28号的11.8版本中被更新修复，比较可惜的是，一次更新修复了全部的漏洞逻辑，不禁令人惊叹。

今天就一起来看看整个漏洞的逻辑~

其实如果关注过通达OA的朋友，应该都会知道通达OA是一个特别庞杂的OA系统，整个系统涉及到2万多个PHP文件，其中除了能访问到的Web逻辑以外，OA还内置了特别多的其他功能，可能是用于定制版的OA，也可能压根就是逻辑太多就有很多忘记了。这里的这个漏洞就特别的朴素。

```
mobile\auth_mobi.php
```



这里可以注意到左边传入的uid会直接从数据库查询，然后查询出来的session id会被直接赋值给当前用户。

也就是说，如果当前站点有正在登录的用户，我们通过遍历uid就可以登录所有当前在线的账号。（唯一的问题是，通达OA有自动掉线机制，不过OA系统有在线用户都很正常）

这里的修复方案也很奇怪，可以注意看上图中右边就是11.8的代码，这段代码直接就被删除了...

在通达OA中，其实涉及到上传文件的地方并不少，而且后台本身就有上传文件的功能，但是通达OA在这方面做的比较好，它设计了两个限制给文件上传。

首先我们关注上传文件的逻辑，主要函数为`td_copy`和`td_move_uploaded_file`这两个，这里我们先关注`td_move_uploaded_file`

```
function td_move_uploaded_file($filename, $destination)
{
    if (!is_uploadable($destination, true)) {
        Message(_("禁止"), _("禁止创建此类型文件"));
        Button_Back();
        exit();
    }
    return move_uploaded_file($filename, $destination);
}
```

这里我们直接跟进`is_uploadable`函数

```
function is_uploadable($FILE_NAME, $checkpath, $func_name)
{
    $EXT_NAME = "";
    $POS = strrpos($FILE_NAME, ".");

    if ($POS === false) {
        $EXT_NAME = $FILE_NAME;
    }
    else {
        $EXT_NAME = strtolower(substr($FILE_NAME, $POS + 1));
        $EXT_NAME = filename_valid($EXT_NAME);

        if ((td_trim($EXT_NAME) == "") || (td_trim(strtolower(substr($EXT_NAME,
0, 3))) == "php")) {
            return false;
        }
    }

    if ($checkpath && !td_path_valid($FILE_NAME, $func_name)) {
        return false;
    }

    ...
```

可以关注到的是：

1、不能没有`.`
2、不能`.`之后为空
3、`.`之后3个字符不能是PHP

第一反应是可以用phtml或者pht等绕过，但可惜通达内置的nginx在这方面配置的很好。

```
        location ~ \.php$ {
            fastcgi_pass   OfficeFPM;
            fastcgi_index  index.php;
            include        fastcgi.conf;

            add_header X-Frame-Options SAMEORIGIN;
        }
```

首先避免了奇奇怪怪的文件后缀，只有php才解析执行。

其次通达还配置了专门的附件目录

```
        location /attachment {
            deny all;
        }
```

一般来说，除非找到绕过的办法，否则所有的文件都会被上传到这个目录下，那么无论我们是否能绕过后缀限制，我们都没办法解析执行php文件。

所以这里我们需要找到一个配合目录穿越的文件上传点。

```
/general/reportshop/utils/upload.php line 116

    else {
        $uploaddir = MYOA_ATTACH_PATH . "reportshop/attachment/";

        if (!is_dir(MYOA_ATTACH_PATH . "reportshop/attachment")) {
            if (!is_dir(MYOA_ATTACH_PATH . "reportshop")) {
                mkdir(MYOA_ATTACH_PATH . "reportshop");
            }

            mkdir(MYOA_ATTACH_PATH . "reportshop/attachment");
        }

        $s_code = mb_detect_encoding($filename, array("UTF-8"), true);

        if ($s_code == "UTF-8") {
            $filename = iconv("UTF-8", "GBK//IGNORE", $filename);
        }

        if (isset($rid)) {
            if (!preg_match("/^\{([0-9A-Z]|-){36}\}$/", $rid) || preg_match("/^\
{([0-9A-Z]|-){36}\}$/", $cid)) {
                if (isset($json)) {
                    echo "{";
                    echo "new_name:'',\n";
                    echo "error: 'true',\n";
                    echo "msg: '文件不符合要求'\n";
                    echo "}";
                }
                else {
                    echo "文件不符合要求! ";
                }

                exit();
            }

            if (!check_filename($filename) || !check_filetype($filename)) {
                if (isset($json)) {
                    echo "{";
                    echo "new_name:'',\n";
                    echo "error: 'true',\n";
                    echo "msg: '文件不符合要求'\n";
                    echo "}";
                }
                else {
                    echo "文件不符合要求! ";
                }
```

```
                exit();
            }

            if (file_exists($uploaddir . $filename)) {
                $p = strpos($filename, "_");
                $s = substr($filename, $p + 1, strlen($filename) - $p - 1);
                $s_prefix = str_replace("-", "", str_replace("}", "",
str_replace("{", "", $rid . $cid)));

                if ($filename != "{" . $s_prefix . "}_" . $s) {
                    td_copy($uploaddir . $filename, "$uploaddir{" . $s_prefix .
"}_" . $s);
                }
            }
        }
        else if (!empty($_FILES)) {
            $s_n = $_FILES[$fileid]["name"];
            if (!check_filename($s_n) || !check_filetype($s_n)) {
                if (isset($json)) {
                    echo "{";
                    echo "new_name:'',\n";
                    echo "error: 'true',\n";
                    echo "msg: '文件不符合要求'\n";
                    echo "}";
                }
                else {
                    echo "文件不符合要求! ";
                }

                exit();
            }

            if (($s_n[0] != "{") && isset($newid)) {
                $s_n = "{" . $newid . "}_" . $s_n;
            }

            if (td_move_uploaded_file($_FILES[$fileid]["tmp_name"], $uploaddir .
$s_n)) {
            }
            else {
                $b_res = "false";
            }
```

首先176行过滤非常简单，后缀不是对应的几个就行了，这里其实过滤没啥用，有很多过滤方式，php5啊，php.，php::$DATA都可以绕过

```
function check_filetype($s_name)
{
    $p = strrpos($s_name, ".");

    if ($p !== false) {
        $postfix = strtolower(substr($s_name, $p + 1));

        if (in_array($postfix, array("php", "exe", "js"))) {
            return false;
        }
    }
```

```
    return true;
}
```

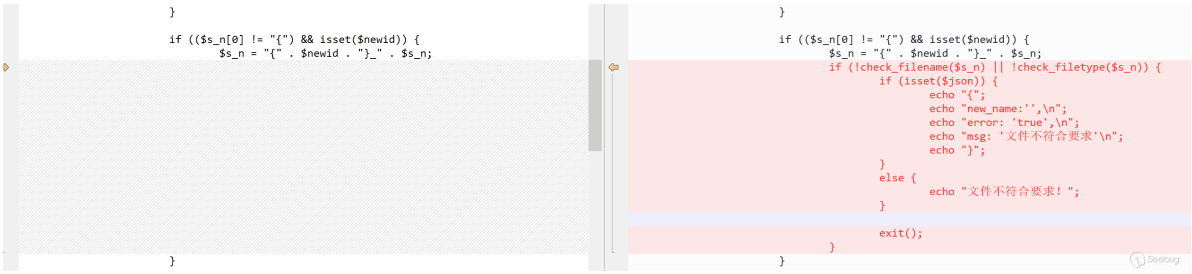
绕过的重心主要在这几行

```
if (($s_n[0] != "{") && isset($newid)) {
    $s_n = "{" . $newid . "}_" . $s_n;
}

if (td_move_uploaded_file($_FILES[$fileid]["tmp_name"], $uploaddir . $s_n)) {
}
else {
    $b_res = "false";
}
```

这里可以关注到newid被直接拼接进了路径中，且没有设计专门的过滤，导致我们可以穿越任意目录写，当newid为`../../../../`目录相应就为

```
D:/MYOA/webroot/attachment/reportshop/attachment/{321/../../../../a}_.txt
```

这里的修复方式也很直接，newid被添加了额外的过滤。



截至目前为止，我们可以将一个非php文件传到任意为止了。在这里曾经困扰了我很久，因为这里的文件上传实际上受到了3个以上的限制，且所有的限制都集中在php后缀，但这里明显是个不太现实的目标。所以与其继续去研究怎么找一个蹩脚的绕过方式，不如去找一个可以文件包含的地方。这里就用到了之前公开的任意文件包含漏洞，之前的漏洞修复方式主要是限制了`..`和权限。

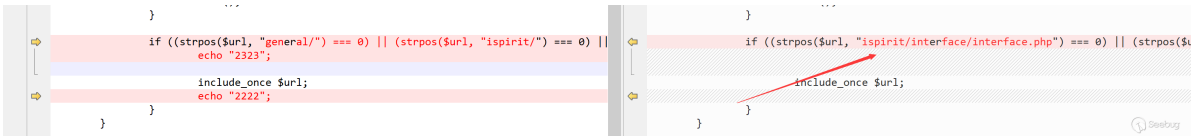
这里我们先看看之前的任意文件包含漏洞。

```
/ispirit/interface/gateway.php

if ($json) {
    $json = stripcslashes($json);
    $json = (array) json_decode($json);

    foreach ($json as $key => $val ) {
        if ($key == "data") {
            $val = (array) $val;

            foreach ($val as $keys => $value ) {
                $keys = $value;
            }
        }

        if ($key == "url") {
            $url = $val;
        }
```

```
    }

    if ($url != "") {
        if (substr($url, 0, 1) == "/") {
            $url = substr($url, 1);
        }

        if (strpos($url, "..") !== false) {
            echo _("ERROR URL");
            exit();
        }

        if ((strpos($url, "general/") === 0) || (strpos($url, "ispirit/") === 0)
|| (strpos($url, "module/") === 0)) {
            include_once $url;
        }
    }
```

可以看到这里限制了general、ispirit、module开头，且添加了专门的过滤，过滤了`..`等目录穿越符号。

这里简单了，我们直接上传txt文件到general目录下，然后包含即可。



这里也可以看到，在11.8版本中，这个文件包含被直接改成指向文件的了。

这个组合漏洞最早是我在2020年年初挖的，一直存在手里也没用上，没想到突然就更新了，修复方式还特别像是翻着漏洞文档一行一行修复的，就感觉很无奈。

其实之前通达OA的安全性一直受人诟病，在11.6开始，逐渐加入全局过滤，然后nginx的配置也经过很多次更新，比较关键的任意用户登录又一再修复，其实后台的漏洞都无关紧要了，这也能说明通达的安全人员也是下了一番苦工的~