Soluzione

Passaggi:

1. **Identifica la vulnerabilità di Command Injection**:
    - L'applicazione permette all'utente di inserire un valore per il parametro `hostname` da inviare al comando `ping`.
    - Nonostante ci sia una sanitizzazione basilare, puoi aggirare i pattern vietati sfruttando caratteri come `${IFS}` per bypassare i controlli di input.
2. **Sfrutta la vulnerabilità di Command Injection**:

Utilizza il seguente input per iniettare un comando maligno e visualizzare il contenuto del file `flag.txt`:
bash
Copy code
```
google;c''at${IFS}./f''lag.txt
```

    -
        - Qui, `${IFS}` rappresenta il carattere di spazio (`space`), che è un separatore di input in ambiente Unix, mentre il comando `cat` viene mascherato con due `'` per evitare il rilevamento.
        - Questo comando eseguirà il `ping` all'host `google`, dopodiché eseguirà `cat ./flag.txt` per leggere il file della flag.
3. **Recupera la flag**:
    - Il contenuto del file `flag.txt` verrà mostrato nel browser come output del comando `cat`.

Teoria e Spiegazione dell'Attacco

Command Injection:

- **Command Injection** si verifica quando un'applicazione passa input non verificato direttamente a un comando di sistema. In questo scenario, l'input utente (`hostname`) viene utilizzato direttamente nella costruzione del comando `ping` senza essere sanitizzato correttamente. Ciò consente a un attaccante di inserire comandi arbitrari e farli eseguire dal sistema operativo.

Bypass dei Controlli di Input:

- Anche se l'applicazione tenta di bloccare l'esecuzione di comandi pericolosi (come `cat`, `&`, `|`), è possibile aggirare queste restrizioni utilizzando tecniche di evasione, come l'uso di separatori alternativi (`IFS`), comandi divisi con apici (`c''at`), o concatenazioni.

Uso di `os.popen`:

- L'uso della funzione `os.popen` per eseguire comandi di shell è particolarmente pericoloso quando l'input utente non è completamente verificato. `os.popen` esegue il comando nella shell, il che significa che qualsiasi input passato può eseguire comandi aggiuntivi se non sanitizzato correttamente.

CWE Coinvolti:

- **CWE-77**: Improper Neutralization of Special Elements used in a Command ('Command Injection').
    - L'applicazione non verifica in modo adeguato i caratteri speciali usati nei comandi di sistema, permettendo così l'iniezione di comandi arbitrari.
- **CWE-20**: Improper Input Validation.
    - L'applicazione non filtra correttamente l'input utente, permettendo l'iniezione di caratteri pericolosi che compromettono la sicurezza del sistema.
- **CWE-88**: Argument Injection or Modification.
    - Gli argomenti passati al comando `ping` possono essere manipolati per eseguire altri comandi arbitrari.