SECURITY ASSESSMENT REPORT

Task 3: Secure Coding Review
Target: Network Diagnostics API

Prepared by: Badar Ijaz
Date: December 30, 2025
Status: FINAL

1. Executive Summary

A comprehensive security audit was conducted on the 'Network Diagnostics API'. The assessment followed a 'Test-Driven Security' methodology, combining static analysis with dynamic exploitation proof-of-concepts.

Vulnerability ID	Issue Type	Severity	Status
VULN-001	OS Command Injection	**CRITICAL (9.8)**	**REMEDIATED**

2. Methodology

The audit followed a four-step lifecycle:

1. Asset Creation: Deployed a local Flask API environment.
2. Assessment: Used 'Bandit' for static analysis and identified CWE-78.
3. Exploitation (Red Team): Developed a Python script to achieve Remote Code Execution (RCE).
4. Remediation (Blue Team): Implemented input allow-listing and secure subprocess execution.

3. Technical Findings & Evidence

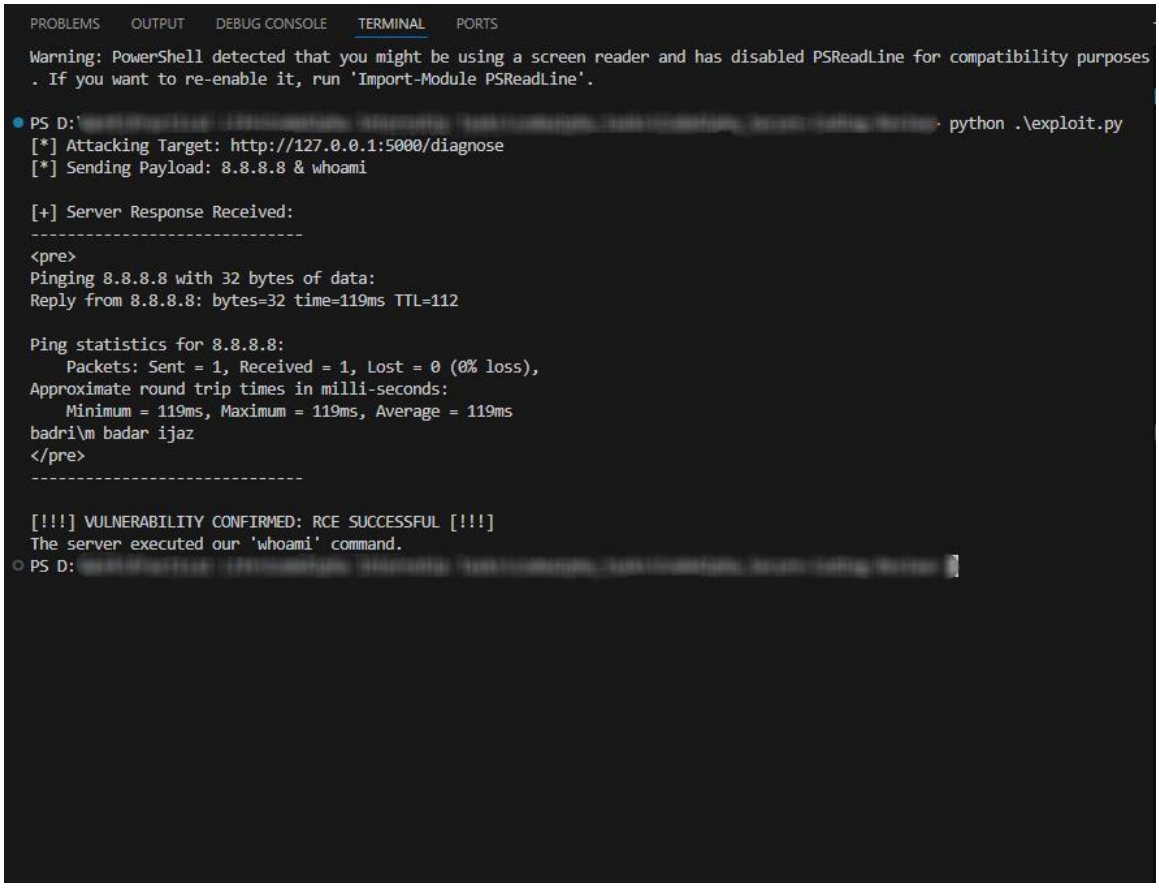
The application failed to sanitize user input before passing it to the system shell.

Vulnerable Code Snippet:

```
command = f"ping -n 1 {target_ip}"
subprocess.check_output(command, shell=True)
```

Proof of Concept (Evidence):

The following screenshot demonstrates the successful execution of the 'whoami' command via the exploit script:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes
. If you want to re-enable it, run 'Import-Module PSReadLine'.
PS D:\ python .\exploit.py
[*] Attacking Target: http://127.0.0.1:5000/diagnose
[*] Sending Payload: 8.8.8.8 & whoami
[+] Server Response Received:
<pre>
Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=119ms TTL=112
Ping statistics for 8.8.8.8:
Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 119ms, Maximum = 119ms, Average = 119ms
badri\m badar ijaz
</pre>
[!!!] VULNERABILITY CONFIRMED: RCE SUCCESSFUL [!!!]
The server executed our 'whoami' command.
PS D:

4. Remediation Strategy

The vulnerability was patched by implementing two controls:

- 1. Disabled Shell Execution (shell=False).
- 2. Implemented strict Allow-List Input Validation.

Secure Code Implementation:

```
# 1. Validation
allowed = set("abcdefghijklmnopqrstuvwxyz0123456789.-")
if not set(ip).issubset(allowed): abort(403)

# 2. Secure Execution (List format)
subprocess.check_output(["ping", "-n", "1", ip], shell=False)
```

5. Verification (Regression Test)

The exploit script was re-executed against the patched server. The server correctly identified the malicious payload and blocked the request (HTTP 403).