week 5 task 1

Q1. python ping_service.py 1.1.1.1; cat /etc/passwd allows to read the contents of the passwd and prints it.

Q2. shell=true line without any sanitazion or validation allows for the attacker to execute code in to the command string that is passed in to the system shell just like in question 1. In this case it should only validate characters 0-9 and ".", because it is asking for the ip address. Also it should sanitize user input to prevent xss or sql injections by stripping user-submitted markup code.

Q3. you can fix them by implementing proper input validation and sanitisation.

Q4. This is how I did the input sanitisation.

```python
import argparse
import subprocess
import re

def pingdevice(target):
    sanitizedtarget = re.sub(r'[^0-9.]', '', target)
    command = f"ping -c 4 {sanitizedtarget}"
    try:
        output = subprocess.checkoutput(command, shell=True, stderr=subprocess.STDOUT)
        print(output.decode())
    except subprocess.CalledProcessError as e:
        print(f"Failed to ping {sanitized_target}\n{e.output.decode()}")

def main():
    parser = argparse.ArgumentParser(description="Ping a device. WARNING: This tool now mitigates command injection.")
    parser.add_argument("target", type=str, help="IP address or DNS name of the target device")
    args = parser.parse_args()
    pingdevice(args.target)

if __name__ == "__main__":
    main()
```

Q5. You can never be 100 percent sure that injection is not possible anymore.