

SHIVESH OJHA

Code Injection in Python

Vulnerable Code –

```
#!/usr/bin/env python3

def inp():
    Calc = input("Enter an arithmetic operation: ")
    return Calc

t = inp()

if not t:
    print("No Input!")
else:
    print("Answer =",eval(t))
```

Output –

```
C:\Users\Shivesh Ojha\Desktop\PDC-codes>test.py
Enter an arithmetic operation: 5+8
Answer = 13
```

The above output is for intended usage.

```
C:\Users\Shivesh Ojha\Desktop\PDC-codes>test.py
Enter an arithmetic operation: eval('__import__("subprocess").getoutput("whoami")')
Answer = laptop-bsd3crq4\shivesh ojha

C:\Users\Shivesh Ojha\Desktop\PDC-codes>test.py
Enter an arithmetic operation: eval('__import__("subprocess").getoutput("net user")')
Answer =
User accounts for \\LAPTOP-bsd3crq4

-----
18BCI0137      Administrator      DefaultAccount
Guest          shive             Shivesh Ojha
University     WDAGUtilityAccount
The command completed successfully.

C:\Users\Shivesh Ojha\Desktop\PDC-codes>
```

The above output is not intended and showcases a vulnerability.

Mitigated Code –

```
#!/usr/bin/env python3

import re

def inp():
    Calc = input("Enter an arithmetic operation: ")
    return Calc

def reg(s):
    ad = "(\\d+)\\+(\\d+)"
    sb = "(\\d+)\\-(\\d+)"
    ml = "(\\d+)\\*(\\d+)"
    div = "(\\d+)\\/(\\d+)"
    x = False
    if(re.search(ad,s) or re.search(sb,s) or re.search(ml,s) or re.search(div,
s) ):
        x = True
    return x

t = inp()

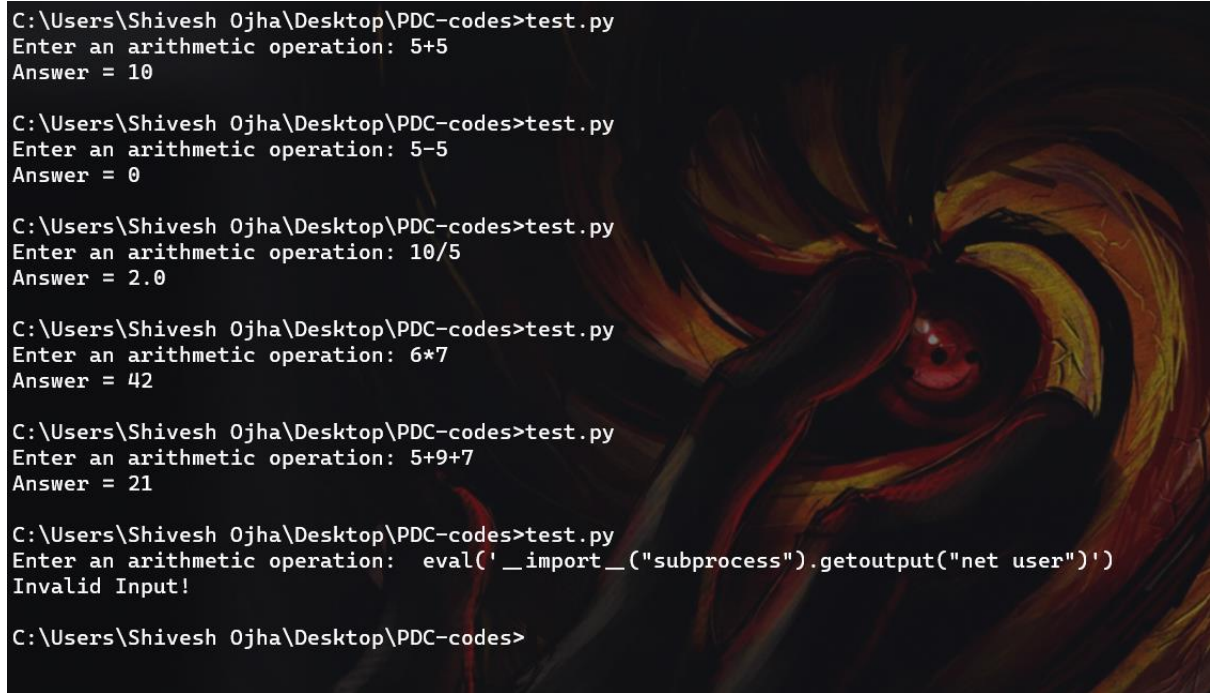
if not t:
    print("No Input!")
elif(reg(t)):
    print("Answer =",eval(t))
else:
    print("Invalid Input!")
```

In the above code, I've added a reg function to validate the input and check if it is an arithmetic operation or not.

It will return True or False based on the validation of the user input.

Thus, helping to mitigate the code injection vulnerability.

Output –



```
C:\Users\Shivesh Ojha\Desktop\PDC-codes>test.py
Enter an arithmetic operation: 5+5
Answer = 10

C:\Users\Shivesh Ojha\Desktop\PDC-codes>test.py
Enter an arithmetic operation: 5-5
Answer = 0

C:\Users\Shivesh Ojha\Desktop\PDC-codes>test.py
Enter an arithmetic operation: 10/5
Answer = 2.0

C:\Users\Shivesh Ojha\Desktop\PDC-codes>test.py
Enter an arithmetic operation: 6*7
Answer = 42

C:\Users\Shivesh Ojha\Desktop\PDC-codes>test.py
Enter an arithmetic operation: 5+9+7
Answer = 21

C:\Users\Shivesh Ojha\Desktop\PDC-codes>test.py
Enter an arithmetic operation: eval('__import__("subprocess").getoutput("net user")')
Invalid Input!

C:\Users\Shivesh Ojha\Desktop\PDC-codes>
```

The last input in the above window shows that it is not accepting values other than arithmetic operation. Thus, successfully mitigating the code injection vulnerability.